Test-Driven Development Using AppleScript—Part 2

MacTech Magazine
May • 2008

MACTECH®

The Journal of Macintosh Technology

The iPhone SDK: Testing the Waters

Learning the Ropes on REALBasic

The Road to Certification: ASCA and ACTC

An Introduction to Core Animation

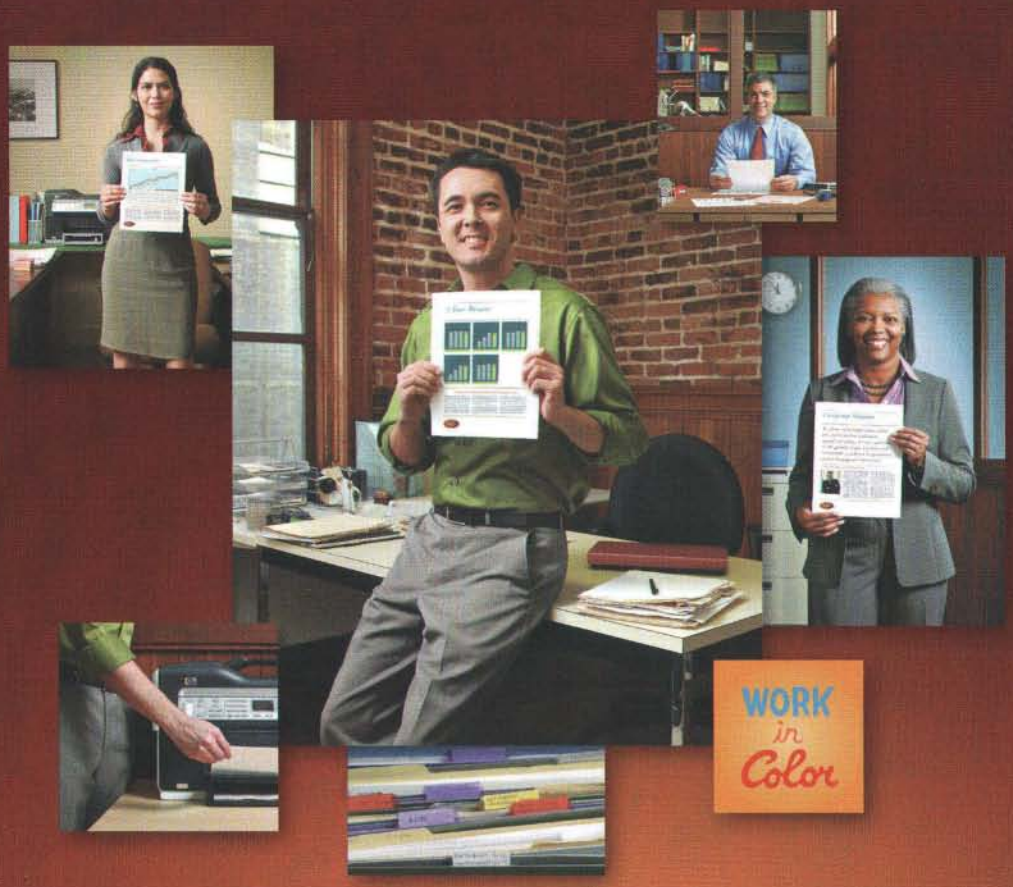
MACTECH.COM

$8.95 US, $12.95 Canada

05

0 32128 74887 8

ISSN 1067-8360 Printed in U.S.A.



UP TO 25% LOWER COST PER PAGE THAN LASER PRINTERS.* HP OFFICEJET PRO ALL-IN-ONES STARTING AT $299.

They fax, scan, copy and deliver laser-quality printing—at an impressively low cost per page. Individual ink cartridges save money too, because you only replace what runs out. HP, *PC Magazine*'s Readers' Choice for service and reliability for 15 years running.

BEST
HP Officejet Pro L7780 AiO.
Plus wireless networking.
$499.

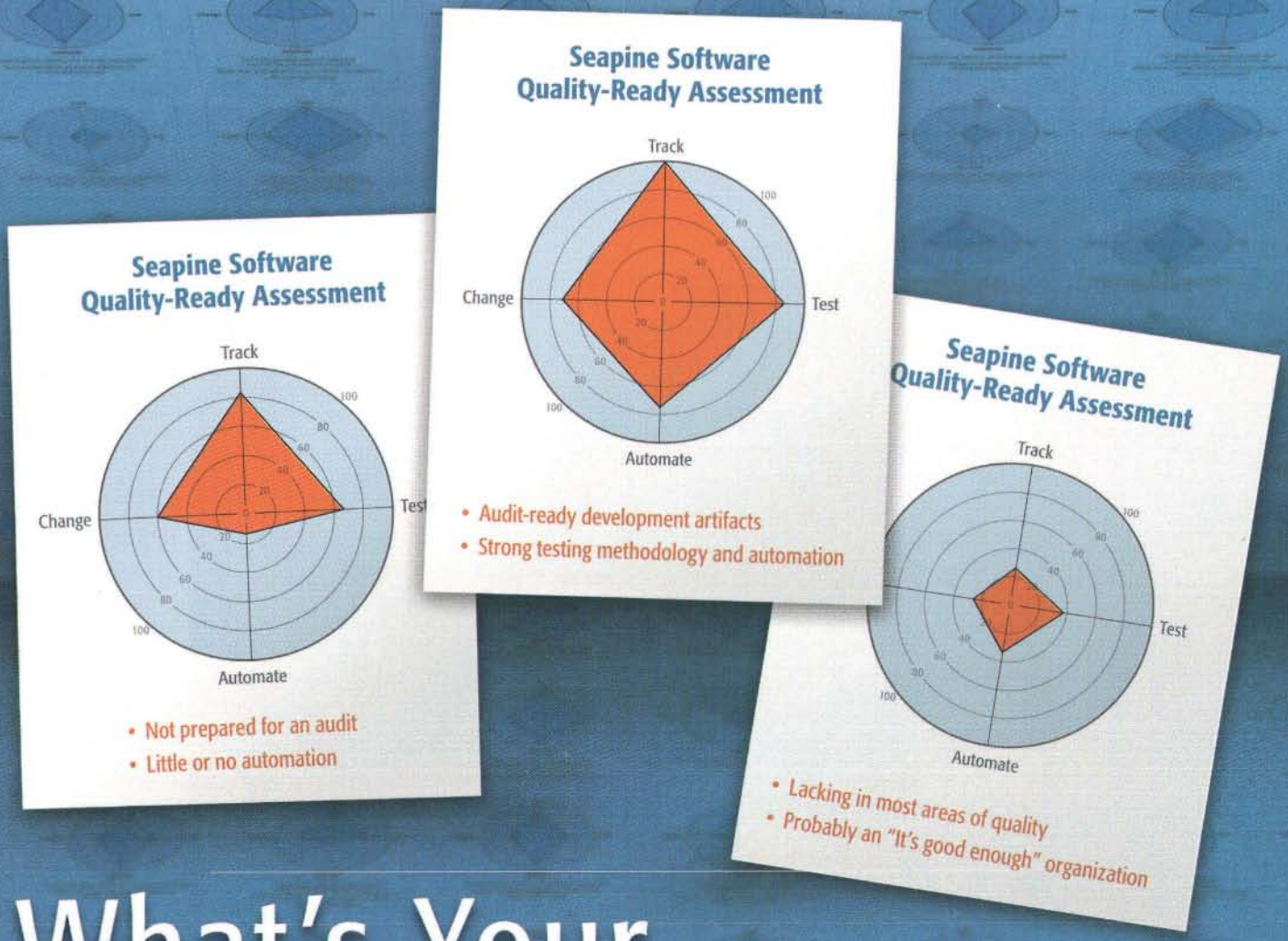
BETTER
HP Officejet Pro L7680 AiO.
Plus two-sided printing.
$399.

GOOD
HP Officejet Pro L7580 AiO.
Up to 35 pages a minute.
$299.

To learn more, visit hp.com/go/doesitall. Call 1-800-888-3119. Or visit your local reseller or retailer.

*Cost per page (CPP) laser supplies comparisons based on published manufacturer specifications of the highest capacity cartridges available for color laser all-in-ones under $1,000 and mono laser all-in-ones under $500, as reported by Current Analysis, Inc. as of July 2006. Officejet Pro L7500/L7600/L7700 AiO CPP based on HP 88 Large Ink Cartridges (not included, sold separately) estimated street price and published yield. Results may vary. Estimated U.S. retail price. Actual price may vary. Simulated images.

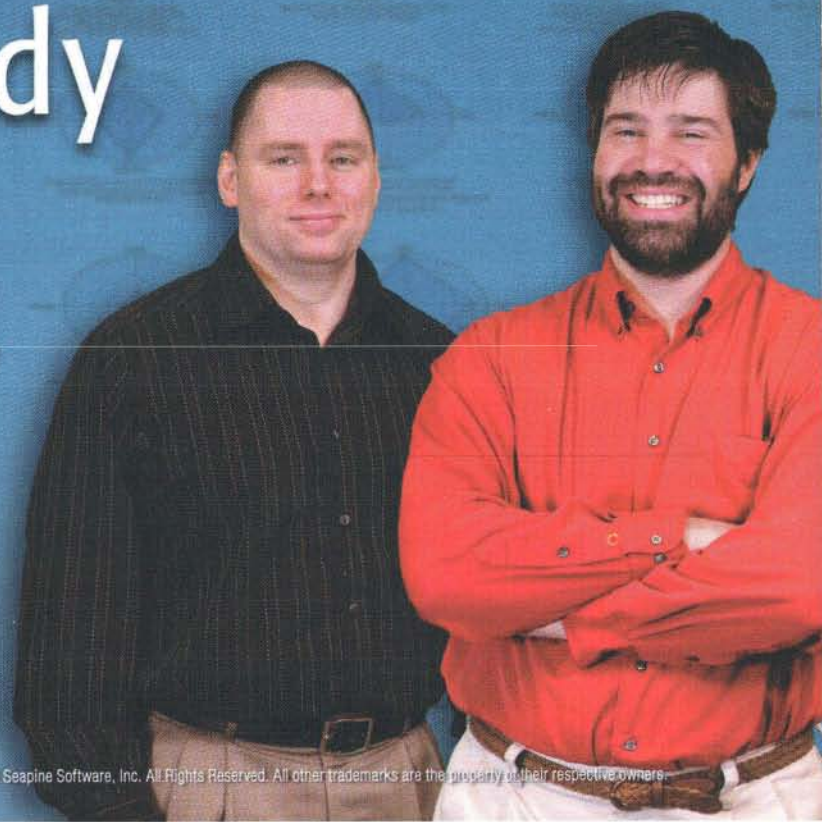
Seapine Software



What's Your Quality-Ready Score?

You need the right tools to build a quality-ready advantage. Take the Seapine Software Quality-Ready Assessment to measure your development and QA teams' ability to create a sustainable competitive advantage through integrated ALM and learn how to make your organization's productivity soar.

Find out your score at **www.seapine.com/qra**



MacMall

MacMall is the #1 Apple Direct Reseller &

All-new processor te

NEW MacBook Pro with multi-touch trackpad.

NEW 15" & 17" MacBook™ Pro

- FASTER! Up to 2.6GHz Intel Core 2 Duo processor with 45nm process technology
- NEW! Up to 6MB processor cache ■ NEW! Multi-touch trackpad
- NEW! 17" model with arsenic-free glass and optional mercury-free LED backlit display

~~$1994~~ – $150 mail-in rebate*
MacBook Pro starting at **$1844!*** #7405626 *Finance for $60/mo.*

NEW MacBook™ Air!

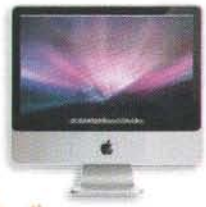
- Up to 1.8GHz Intel Core 2 Duo processor ■ Only 0.76" thin
- 80GB hard drive or 64GB solid-state drive ■ Weighs just 3lbs.!
- 802.11n wireless ■ 13.3" glossy widescreen backlit display
- Full-sized QWERTY keyboard ■ Built-in iSight™ video camera

~~$1794~~ – $50 mail-in rebate
starting at **$1744!*** #7373085

NEW Mac® Pro!

- NEW! Up to 8-core performance with up to two Quad-Core Intel Xeon 5400 series processors at up to 3.2GHz
- NEW! ATI Radeon HD 2600 XT graphics with 256MB GDDR3
- NEW! Four internal drive bays support up to 4TB of storage

~~$2289~~ – $75 mail-in rebate
starting at **$2214!*** #7371990



Apple iMac®!

- Intel Core 2 Duo processor at up to 2.4GHz or Intel Core 2 Extreme processor at 2.8GHz ■ Up to 500GB hard drive
- 20" or 24" glossy widescreen display ■ ATI Radeon HD graphics
- Built-in iSight®, wireless and Gigabit Ethernet

~~$1194~~ – $50 mail-in rebate
starting at **$1144!*** #7288171

VMware Fusion for Mac
#7288411

~~$67.99~~ **$17.99!**
After $50 mfr. mail-in rebate. Price before rebate is $67.99. See below.*

Parallels Desktop 3.0 for Mac
#7257816

FREE!
After rebate with Apple computer purchase. See below.*

Apple Aperture® 2.0 Upgrade from 1.0 or 1.5
#7388601

upgrade **$89.98!**

NEW! Microsoft Office 2008!

New versions of your favorite office applications with support for both Intel and PowerPC Macs!

Microsoft Office 2008 for Mac Home and Student Edition
only **$129.98!** #7352258

Allows for three installations from one box!

*WE'LL BEAT ANY PRICE ON ANY APPLE BRAND PRODUCTS or competitive promotion on any Apple computer. We may require proof of dealer's pricing. Competitor must be an Apple Authorized Reseller. Product must be in stock at the competition. Advertised item must be identical to the one offered by MacMall and must be a new item in a factory sealed box. We reserve the right to limit quantity ordered. This offer does not apply to special, bonus or free offers, grand openings, special purchases or special buys, manufacturer's rebates, closeouts, clearances or to dealers' one-of-a-kind or other limited quantity offers. Nor does it apply to advertising errors made by any authorized dealer. Not open to dealers. Void where prohibited. Sorry, offer not valid in conjunction with other rebate offers. On multiple-item quotes: "We'll Beat Any Price" applies to the price of the entire order, not individual line items. Offer applies to pre-taxed prices only. Offer valid only at time of purchase, not after the fact. ■ UP TO $200 CASH BACK OFFER-Get up to $150 Cash Back via MacMall mail-in rebate with purchase of select computer models. Ends 4/23/08. ■ FREE PRINTER OFFER-Get an Epson Stylus C120 Color Inkjet Printer FREE after $50 and $29.99 mfr. mail-in rebates with purchase of any Apple computer. Price before

Authorized Reseller

the Mac pro's choice!

chnologies!

MacMall Exclusive Deals!
Get up to $200 cash back!*
FREE Parallels Desktop 3.0!*
FREE Epson Inkjet Printer!*
See below for details.
MacMall is your source for LCD and Plasma TVs!

The NEW MacBook runs applications faster for the same low price.

- FASTER! Up to 2.4GHz Intel Core 2 Duo processor with NEW 45nm process technology
- NEW! Up to 250GB hard disk drive
- NEW! Up to 2GB DDR2 SDRAM; exp. to 4GB
- 13.3" glossy widescreen TFT display
- Slot-loading SuperDrive or DVD/CD-RW combo drive
- AirPort® Extreme 802.11n Wi-Fi wireless
- Bluetooth 2.0 + EDR ■ Gigabit Ethernet networking
- Built-in iSight® video conferencing and omnidirectional microphone
- iLife® '08 software suite ■ Mac OS® X 10.5 Leopard™

~~$1094~~ – $75 mail-in rebate
NEW MacBook starting at
$1019!* #7404921
Finance for $33/mo.

1-800-622-6255 macmall.com Source Code: MACTECH

Your iPod Superstore!

With every flavor of player and the best special offers your can find, MacMall is your source for everything iPod! And you can visit us online for the widest selection of accessories at the best prices!

Call for iPods in all colors and capacities!

#7297622	1GB iPod® shuffle™ (Blue)	new low price $47.99
#7404637	NEW! 2GB iPod® shuffle™ (Green)	only $67.99
#7297625	iPod® nano (Silver)	starting at $144
#7297631	iPod® classic (Silver)	starting at $244

**Free case does not apply to iPod shuffle.

NEW iPod® touch!

32GB now available!

- Includes NEW software revision 1.1.3 with Mail, Maps, Stocks, Weather and Notes ■ Features Wi-Fi Web browsing
- Multi-touch interface ■ View movies, photos and YouTube videos
- Visit the iTunes Wi-Fi Music Store from any wireless hotspot

Check our Web site for best price!

#7373111	iPod® touch 8GB	$294
#7373112	iPod® touch 16GB	$394
#7387505	NEW! iPod® touch 32GB	$494

Closeout Specials on MacBook Pro!

15.4" MacBook Pro—2.2GHz/2GB/120GB HD
~~$1694~~ – $150 mail-in rebate
starting at **$1544!*** #7254324

15.4" MacBook Pro—2.2GHz/2GB/160GB HD
~~$1894~~ – $150 mail-in rebate
starting at **$1744!*** #7254325

Adobe Creative Suite 3 Master Collection
Call for affordable upgrade paths!
#7228222
only **$2398.49!**

Apple Cinema Displays

#459836	20" Cinema Display®	*was ~~$624~~ now* $594
#459837	23" Cinema HD Display	*was ~~$994~~ now* $894
#459838	30" Cinema HD Display	*was ~~$1994~~ now* $1794

Apple® AirPort® Extreme Base Station
802.11a/b/g/n
#7288377
only **$173.99!**

Mac OS® X Leopard™
#7355254
only **$109.98!**
#7355255 Family 5-Pack $179.99

rebates is $79.99. Ends 4/23/08. ■ FUSION OFFER-Price after $50 mfr. mail-in rebate with purchase of any Apple CPU or Mac OS X Leopard. Price before rebate is $67.99. Offer expires 6/30/08. ■ FREE PARALLELS DESKTOP 3.0 OFFER-Get Parallels Desktop 3.0 for Mac (#7257816) FREE after $20 mfr. and $60 MacMall mail-in rebates with purchase of any new Intel-based Apple CPU. Price before rebates is $80. Ends 4/23/08. ■ FREE CASE OFFER-After $5.99 MacMall mail-in rebate with purchase of any iPod touch, iPod classic or iPod nano. Price before rebate is $5.99. Ends 4/23/08. ■ FREE ENGRAVING OFFER-Get select touch, iPod classic, iPod nano and iPod shuffle models engraved for FREE when you purchase them through MacMall. No rebate required. There is a $9.99 charge for select iPods. Call or visit www.macmall.com/myipod to place your order. ■ ALL OFFERS VALID ONLY WHILE SUPPLIES LAST. ■ †For rebate terms and conditions, please visit our Web site and enter the applicable part number. Download rebate coupons at www.macmall.com/rebates.

It's always nice to have multiple options.

NEW!

*HASP, the #1 Choice of Software Developers Worldwide**

**Source: Frost & Sullivan #N1AF-70, IDC Bulletin #34452*

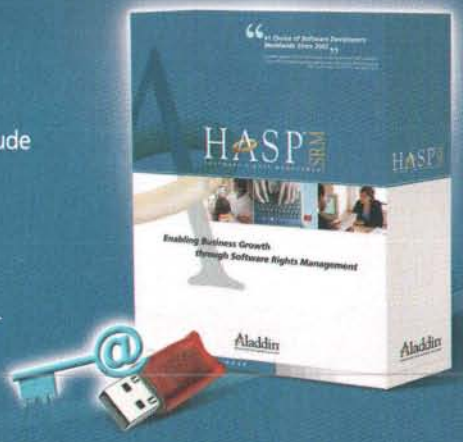
HASP® SRM
SOFTWARE RIGHTS MANAGEMENT

Get the perfect fit! Hardware- and software-based protection, licensing and distribution – all from a single solution!

HASP SRM is a powerful software rights management solution enabling you to secure revenue, maintain competitive advantage and accelerate business growth. HASP SRM gives you a multitude of options for:

- *Strong IP & software anti-piracy protection*
- *Secure & flexible licensing and distribution*
- *Hardware- and software-based protection keys*

Discover how a truly role-based solution reduces the cost of managing the DRM lifecycle. Choose licensing models and protection keys completely independent of the protection process – at any time throughout the software lifecycle. Advanced, but easy to use, HASP SRM delivers enhanced marketing power and takes software rights management to a whole new level!



Request a FREE HASP SRM Developer Kit at www.Aladdin.com/MT8

• NORTH AMERICA: 1-800-562-2543, 847-818-3800
• UK • GERMANY • ISRAEL • BENELUX • FRANCE • SPAIN • ITALY • INDIA • CHINA • JAPAN

Aladdin
SECURING THE GLOBAL VILLAGE

2007 FROST & SULLIVAN Global Product Innovation Award
Frost & Sullivan 2007 Product Innovation Award Software DRM

2008 SIIA //CODiE// FINALIST
SIIA CODiE Awards 2008 Finalist Software DRM

www.Aladdin.com/HASP

ARTICLES & DEPARTMENTS

From the Editor

There are many interesting things happening in the Apple world today: iPhone improvements, Leopard improvements, better availability of community tools that help everyone, and more. My mind has been drawn to all of them on some level, and that's being borne out in the pages of MacTech. Like many wisdom traditions, the tech world really operates on two levels. One is the heads-down things that interest you at the moment. You need to establish a hierarchy of Open Directory Replicas that serve 12 disparate offices. You need to solve an I/O issue on a server, or, you need to add some nifty CoreAnimation effects to an application. The second level is The Big Picture. Each of the level one projects are part of a greater whole. What happens in one realm affects the other. More importantly, understanding in one realm *improves* the work you do in others.

To this effect, we've been running some more "developer-ish" type of content. But the sysadmin content lives on. Or, are they both part of a greater whole? If you're a developer, your application writes to a server. Those reads are writes are subject to permissions on the server – something we talk about often. Conversely, if you're a sysadmin, and are troubleshooting a problem, it's great to be able to read the source that Apple provides us (see http://developer.apple.com/opensource). Even better: it's nice – if not essential – to be able to provide custom solutions for your environment. This is done through programming. Call it scripting, call it development, or call it being learned: *it's all a part of the greater technical whole.*

On to this month's issue. On the cover: the iPhone SDK. At the time of this writing, the bulk of the SDK is still under NDA, and we'll be bringing you greater coverage when we're able. For the time being, **Rich Warren** gets you up to speed with the current state of the **iPhone SDK**.

Rich also brings up part 2 of his **RubyCocoa** article. Apple is making a bold statement by making Ruby and other "scripting" languages available in Xcode and able to access Cocoa calls. The bar has been lowered (or expanded, depending on your point of view) for full, Mac-native, application development.

Doug Hanley fills in the final piece of the **Leopard certification** puzzle. Part 5: ACSA and ACTC was slightly delayed as parts of the course were still being decided on when it was initially to be written. Those details have been worked out, and Doug has the details for you.

Repeatable testing, or Unit Testing, is a hot topic. Justifiably so. Automated tests help to ensure that changes and updates don't break something else along the way. **Andy Sylvester** brings us the second part of **Test-Driven Development Using AppleScript**. Learn to confirm updates using ASUnit.

Dave Dribin once again leads us down **The Road to Code**. If you've been following along so far, you're probably pretty capable about now. Of course, there's always more to learn! This month, Dave dives deeper into some more Cocoa intricacies.

For those of you feeling a little more advanced, **Marcus Zarra** talks about **CoreAnimation**. Learn to add this amazing new technology to your Cocoa application!

Greg Neagle writes this month's **MacEnterprise** column, focusing on **Managed Preferences** for mobile machines. This is an oft-requested topic, and Greg spells it out nicely!

Norman Palardy introduces us to **REALBasic**, the first in a series to get people up to speed with this wonderfully useful development environment. Again, we have a tool that lowers the bar to creating customized, native OS X solutions. REALBasic provides an integrated IDE and debugger, along with methods to tap into Cocoa API calls. Easy to get into, but powerful enough to write as sophisticated an application as you could imagine.

This month's **Mac In The Shell** wraps up coverage of PHP as a generic scripting language. The promised file manipulation techniques and more are presented.

Last, but not least: Check out Alexander Schoen and Peter Maurer of Many Tricks in this month's MacTech Spotlight. Many Tricks produce the popular yFlicks, Which and Desktop Curtain applications...and more, so, give 'em a look!

Finally, WWDC time is nigh! We hope to see everyone in San Franciso for what promises to be a very exciting conference. Until next month, enjoy!

Ed Marczak,
Executive Editor

Communicate With Us

Department E-Mails

Orders, Circulation, & Customer Service
cust_service@mactech.com

Press Releases
press_releases@mactech.com

Ad Sales
adsales@mactech.com

Editorial
editorial@mactech.com
(Authors only, no pr)

Accounting
accounting@mactech.com

Marketing
marketing@mactech.com

General
info@mactech.com

Web Site
http://www.mactech.com

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail?**

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact MacTech Magazine Customer Service at 877-MACTECH

We love to hear from you! Please feel free to contact us with any suggestions or questions at any time.

Write to letters@mactech.com or editorial@mactech.com as appropriate.

MACTECH®
The Journal of Macintosh Technology

A publication of XPLAIN CORPORATION

Canada Post: Publications Mail Agreement #41513541
Canada Returns to be sent to: Bleuchip International, P.O. Box 25542, London, ON N6C 6B2

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are $47.00 per year. Canadian subscriptions are $59.00 per year. All other international subscriptions are $97.00 per year. Domestic source code disk subscriptions are $77 per year. All international disk subscriptions are $97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

RubyCocoa

A new way to write Cocoa applications—Part 2

by Rich Warren

Last time, we used RubyCocoa to build a basic RSS feed reader. Now, let's shift things into high gear. If a feed includes an audible enclosure, we want to download the file and send it to iTunes. We will also explore the debugging options for RubyCocoa, and look at a few cool Ruby tricks.

But, before we get started, there's a little bit of necessary housekeeping. If you haven't done so already, grab the article's source code from ftp.mactech.com/src/mactech/volume24_2008/24.05.sit.

Now, we need to install the RubyOSA library. Again, we can do this using RubyGems. From the command line, type:

```
sudo gem install rubyosa
```

If you remember, last time we had some trouble with the FeedReader library after upgrading OS X to 10.5.2. Well, RubyOSA has its own little problems.

RubyOSA depends on libxml-ruby. OS X comes with version 0.3.8.4 already installed, and RubyOSA should work fine with this. Unfortunately, other RubyGem commands might update libxml-ruby to version 0.5.2.0. This will break RubyOSA.

If you're worried, try `gem list libxml-ruby`. If the command returns `libxml-ruby (0.5.2.0, 0.3.8.4)`, then you have both versions installed. Simply uninstall the 0.5.2.0 version:

```
sudo gem uninstall libxml-ruby -version 0.5.2.0.
```

If all else fails, uninstall both libxml-ruby and RubyOSA. Then reinstall RubyOSA.

Sending Files to iTunes

Go ahead, build and launch the application. Subscribe to an RSS feed. Subscribe to a podcast. Now, click on some of the feeds.

Thanks to the magic of Bindings, the "Send To iTunes" button remains disabled until we select an audio `Enclosure`. We just need to send this enclosure to iTunes. More importantly, we want the file to show up in a RubyRSS playlist. We could do this with AppleScript, but I have issues with AppleScript. I won't go into the gory details here. Suffice it to say, AppleScript knows what it did, and I'm still waiting for an apology.

Fortunately, we have a better way. The RubyOSA library basically wraps AppleScript calls in good old Ruby code.

So, let's get started. First, we want to generate the API documentation for iTunes. Fortunately, RubyOSA comes with a command line tool for building rdoc-style html pages. Make sure iTunes is running. Open Terminal and navigate to the desired directory. Then type the following command:

```
rdoc-osa -name iTunes
```

Give it a few seconds, and you should have the complete iTunes documentation. Unfortunately, I sometimes find the RubyOSA documentation hard to follow. It's a good starting point—but I often find myself firing up the Script Editor and opening the application's Dictionary directly. Between the two, I can usually figure out what's going on.

I also recommend using irb when developing RubyOSA code. If you're not sure what something does. . .well. . .try it out and see. Ruby also lets you inspect the details of all your objects, and you can find a lot of useful information this way. Try this. Launch irb, then enter the following commands:

```
>> require 'rubygems'
=> false
>> require 'rbosa'
=> true
>> app = OSA.app('iTunes')
=> #<OSA::ITunes::Application:0x5b85ec
desc="'sign'($6B6F6F68$)">
>> app.class
=> OSA::ITunes::Application
>> app.methods
=> ["current_encoder", "inspect", "to_yaml_style",
"version", "set", "frontmost?", . . .
```

OK, enough chatter; let's get back to our code.

Hickory Dickory Dock. We Make Your Server Rock.

"We Make Mac Networking Child's Play."

Hello,

We're Small Tree Communications, The Mac networking experts.

Founded in 2003, by a group of talented networking and kernel engineers with high performance supercomputing experience, Small Tree Communications sells and supports high speed networks to make your work-flow run faster at a *surprisingly affordable* price.

- **SAVE TIME on really big file transfers with our across-the-board jumbo frame support**
- **SHARE all your stuff with all your friends off a platform you know and trust: An Apple Xserve**
- **SQUEEZE every bit of bandwidth out of your expensive RAIDS with Small Tree's Link Aggregation Compatible Switches**
- **BAN THE SAN: *Save your money, share your stuff***

Run Faster

"We make all your stuff run faster"

Enter Our Extreme Mac-Over® Contest!

It's so easy, it's child's play.

Send an email to *mac-over@small-tree.com*. In the subject line type *Mac-Over*, and then tell us about your pain, and why you think you deserve a Mac-Over. The winner will receive a *complete consultation, expert advice* and *FREE* Small Tree products to Mac-Over their network.

All entries will receive a discount coupon good for *10% Off* any Small Tree product.

To see all our cool toys, visit us at runfaster.small-tree.com

Small Tree COMMUNICATIONS

Small Tree Communications 7300 Hudson Blvd., Suite 110, Oakdale, MN 55128 866.STC.4MAC email: runfaster@small-tree.com

Again, we should place the RubyOSA code in its own class. Create a new Ruby class named `ItunesSender.rb`. Now type in the following:

ItunesSender.rb

```
require 'rubygems'
require 'rbosa'
require 'open-uri'

# ItunesSender adds a playlist and songs to iTunes using
rbosa.
class ItunesSender

    NAME = 'RubyRSS'

    # add methods here
end
```

The required files should all make sense. We need rubygems to access other gems libraries. RubyOSA lets us make scripting calls on iTunes, and open-uri lets us download the enclosure.

As our one and only public method, `download_files()` will download the file from the provided URL, then it calls `send_file()` on the result.

download_file()

```
# Downloads a file from the given url.
def download_file(url)

    file_name = "#{ENV['HOME']}/Downloads/\
                        #{get_file_name(url)}"

    File.open(file_name, 'w') do |temp|
        open(url) do |file|
            data = file.read
            temp.write(data)
        end
    end


    send_file(file_name)
    File.delete(file_name)

end
```

Now for the helper methods. The `send_file()` method sends the file to the RubyRSS playlist in iTunes. `Get_playlist()` calls `find_playlist()` which searches for the RubyRSS playlist. If `find_playlist()` cannot locate the playlist, `get_playlist()` creates one.

Finally, `get_file_name()` extracts the file's name from its URL.

ItunesSender's Helper Functions

```
private

# Helper Function: adds the given file to the ArticleSandbox
playlist on
# iTunes.
def send_file(file)

    app = OSA.app('iTunes')
    OSA.utf8_strings = true

    playlist = get_playlist(app)

    file = app.add(file, playlist)
end


# Helper Function: returns the ArticleSandbox playlist.
# If the playlist does not exist, this will create it.
def get_playlist(app)

    playlist = find_playlist(app)

    if playlist.nil?
        playlist = app.make(OSA::ITunes::UserPlaylist)
        playlist.name = NAME
    end

    return playlist
end

# Helper Function: finds the existing ArticleSandbox
# playlist. If the playlist does not exist, returns nil.
def find_playlist(app)

    app.sources.each do |source|
        source.playlists.each do |playlist|
            return playlist if playlist.name == NAME
        end
    end

    return nil
end

# Helper Function: extracts the file name from the url.
def get_file_name(url)
    url.split('/').last
end
```

Now we just need to link this into the main controller. Again, initialize the ItunesSender object in the controller's `awakeFromNib()` method.

```
@itunes_sender = ItunesSender.new
```

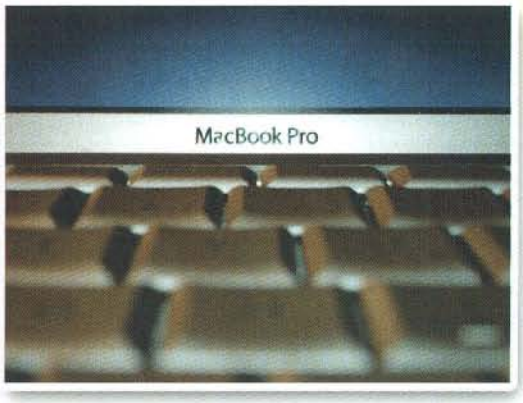
Then update `sendToItunesAction()`.

sendToItunesAction()

```
# This action sends the currently selected Enclosure to
iTunes.

def sendToItunesAction(sender)

    index = @enclosures_table.selectedRow
    return if index < 0


    enclosure = @enclosures.arrangedObjects[index]

    Thread.new do
        begin
            @progress.startAnimation(self)
            @itunes_sender.download_file(enclosure.url)
            @progress.stopAnimation(self)
        rescue Exception => e
            puts e.message
            OSX::NSApp.stop(self)
        end
    end
end
ib_action :sendToItunesAction
```

Short Term Computer Needs?

- Latest Apple Technology
- Legacy Macs
- Windows Computers



- Rent One or Hundreds
- Audiovisual Gear
- Projectors, Plasmas, LCDs
- B&W & Color Copiers

Mac Rentals
Event Technology Services

- Name Brands
- Nationwide Delivery
- Professional Setup
- Onsite Technicians
- Same Day Service

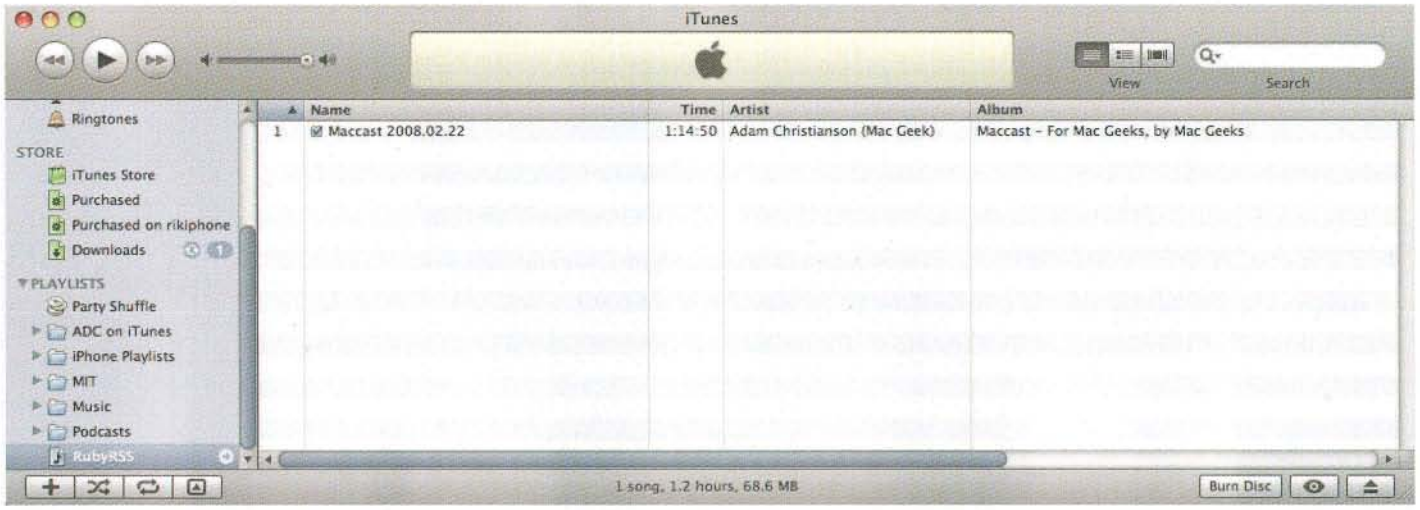
- Compatibility & Compliance Testing
- Short Term Projects, Production Needs
- Trade Shows, Conventions, Corporate Events
- Trainings Sessions, Classrooms
- Fortune 100 and Educational P.O.'s Accepted!

Toll Free
800-756-6227
Outside US/Canada: 858-454-8535

For Online Quote Request, Visit:

www.madcomputing.com

Largest Apple Rental Inventory in the U.S.!



The Complete RubyRSS

Again, this can take a long time, especially when downloading hour-long podcasts. So we spawn off another worker thread. We also reassure our users by animating the progress indicator, and we catch and log any errors that occur during the download.

That's it. We're done (well, for a given value of done). Launch the application, and try sending an enclosure to iTunes. You may need to wait a bit. Go get a cup of copy. Go walk the dog, or re-sort your sock drawer. When you get back, you should find the file in a newly created RubyRSS playlist.

Debugging

Our application looks pretty good, but what do we do when things go wrong? As I mentioned earlier, we cannot use Xcode's built-in debugger (at least, not yet). But that does not leave us helpless in the face of bugs. We still have a number of powerful tools at our fingertips.

First, we could use Ruby's built-in debugger, but there's a better option. Install the ruby-debug gem.

<h1> SPICE UP THAT WEBSITE OF YOURS </h1>

And spice up your relationship with search engines in the process.

Now you do not have to choose between tasteful typography and good results in search engines.

learn why and how at **http://www.photofont.com**

Photofont

More than just one face of the web

```
sudo gem install ruby-debug
```

Now, we can add a breakpoint in our code using the ruby-debug API. Let's add a breakpoint to our main controller's `awakeFromNib()` method. First, import the ruby-debug library.

```
require 'ruby-debug'
```

Next, add a call to `debugger()`. When we execute the code, it will stop at the `debugger()` line, and dump control to the console.

Adding Breakpoints

We have programmatically added a ruby-debug break point to the `awakeFromNib()` method. The `debugger()` call will halt execution and shift control to the debugger.

```
# Initializes the Main Window after it is loaded from the
NIB.
def awakeFromNib

    debugger # this is our breakpoint
    @progress.setDisplayedWhenStopped(false)

    @posts.addObserver_forKeyPath_options_context(self,
"selection",
        0, nil)

    @feed_reader = FeedReader.new(self,
        @app_delegate.managedObjectContext)

    @itunes_sender = ItunesSender.new
end
```

We need to rebuild the application before we can use the debugger. More importantly, we need to launch the application from the command line. Open Terminal, and navigate to the project's directory. Now enter the following command:

```
build/Release/RubyRSS.app/Contents/MacOS/RubyRSS
```

You may need to change "Release" to "Debug", depending on Xcode's configuration. If the command works, you should see something similar to the following:

```
/Users/rikiwarren/Desktop/RubyRSS/build/Release/RubyRSS.app/
Contents/Resources/MainController.rb:24
@progress.setDisplayedWhenStopped(false)
(rdb:1)
```

We have now entered the interactive debugger. Type `help` to get a list of commands.

```
(rdb:1) help
ruby-debug help v0.10.0
Type 'help <command-name>' for help on a specific command

Available commands:
backtrace disable exit    irb    pp     restart step     up
break     display finish  list   ps     save    thread   var
catch     down    frame   method putl   set     tmate    where
```

Record. Edit. Play.

WireTap Studio's patent-pending LivePreview™ technology lets you hear your encoded audio before you record. Lossless master recording technology lets you play with compression settings, effects, and also edit your audio without any loss of quality or content.

It's a true breakthrough in audio recording technology.

WireTap Studio

Download your free trial now: http://www.AmbrosiaSW.com/MacTech

AMBROSIA®
SOFTWARE INC

Macworld Editors' Choice ★★★★★

The Mac Observer Editors Choice Award 2008

Audio Units · Mac · Universal · Works with .mac · WireTap Studio *I'd Tap That!*

Macworld 4½ · 2007 winner macworld eddys

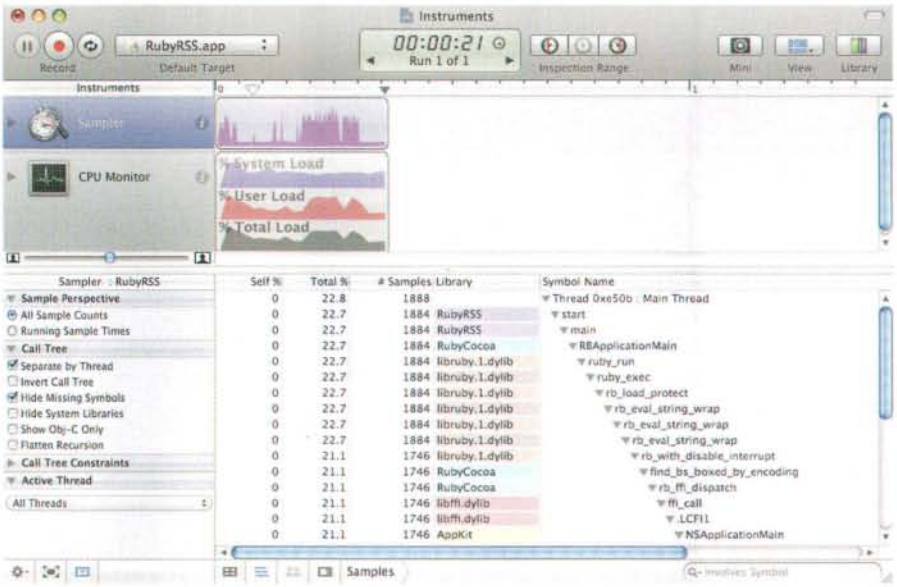
WireTap Studio requires a Mac with an Intel, PowerPC G5, or PowerPC G4 (1GHz or faster) processor and Mac OS X 10.4 or later.

```
continue enable  help    next   quit          show    trace
delete   eval    info    p      reload source  undisplay
```

Cool, yeah. But that's not all. RubyCocoa supports DTrace and Apple's Instruments. This gives us some incredibly powerful tools for analyzing our code.

For example, launch Instruments. Then select the **CPU Sampler** template. Now, click on the **Default Target** drop down menu and select **Launch Executable fi Choose Executable...** Then navigate to our application (in the Project directory at `build/Release/RubyRSS.app`).

Click the **Record** button. When the RubyRSS UI comes up, press the **Refresh Feeds** button. Once finished, go back to Instruments and press **Stop**. That gives us CPU loads and a call tree for our application as it downloads and parses RSS feeds.



Instruments' CPU Sampling

Two Last Cool Tricks

Sure, building a RubyCocoa application from the ground up is great, but what if you just want to add a bit of Ruby to an existing project? Nothing could be simpler. Well. . .OK. . .a lot of things could be simpler. But it's still pretty easy.

RubyCocoa includes an Objective-C function that executes arbitrary Ruby code, `RBBundleInit()`. This function is declared in `/System/Library/Frameworks/RubyCocoa.framework/Headers/RBRuntime.h`, but I've copied it below.

RBBundleInit()

```
int RBBundleInit (const char* path_to_ruby_program,
    Class objc_class, id additional_param);
```

The first argument, obviously, is the path to your Ruby code. The third argument is also easy, `RBBundleInit` simply passes that `id` to your Ruby code as an argument.

But, what about the `objc_class`? Well, here's the good news, you can just set it to `nil`. Just make sure you use an absolute path to your code.

However, if you give it an Objective-C class, RubyCocoa will search for the bundle that contains the given class. It will then append that bundle's Resources directory to Ruby's `$LOAD_PATH`.

If you plan to distribute your code, you will probably want your Ruby code embedded in the application bundle. Just use one of your application's classes for `objc_class`, then copy your Ruby code into the Resources directory.

And, if that trick isn't nearly cool enough. We can also use RubyCocoa to root around in someone else's applications. However, this trick needs a little help from our friends. Download RubyInject from RubyCocoa's Sourceforge page (http://rubycocoa.sourceforge.net/).

Untar the downloaded file (you can just double click its icon in the Finder). Now open Terminal and navigate to the RubyInject folder. You need to build and install the RubyInject framework. Enter these commands:

```
xcodebuild SYMROOT=/tmp
cp -r /tmp/Release/RubyInject.framework
/Library/Frameworks
```

OK, now we just need to run the `inject.rb` command. But before we can do that, we need to get the process id of our target application. Open Activity Monitor and look for the desired application. Process IDs appear in the left-most column.

Now run the following command:

```
sudo ruby inject.rb <pid>
```

This will open up an interactive shell. If you would rather just run a script, then append the script name to the end of the command.

```
sudo ruby inject.rb <pid> <script name>
```

For example, on my computer, iTunes's pid is currently 1014. The following is a sample, interactive session.

Sample RubyInect Session

```
sudo ruby inject.rb 1014
>> require 'osx/cocoa'
=> "true"
>> include OSX
=> "Object"
>> NSApp
=> "#<OSX::NSApplication:0xba0cdd6 class='NSApplication'
```

Premium Small Business Management and Accounting Software

...and now

Point of Sale for Mac
MYOB Checkout

MYOB

Mind Your Own Business. Smarter.

800 322 MYOB (6962)
www.myob-us.com

```
id=0x113ce0>"
>> windows = NSApp.windows
=> "#<OSX::NSCFArray:0xba0365a class='NSCFArray'
id=0x177aac40>"
>> windows.size
=> "3"
>> windows[0].title.to_s
=> "\"Activity Monitor\""
>> windows[1].title.to_s
=> "\"Panel\""
>> windows[2].title.to_s
=> "\"\""
>> ^D%
```

Note, I typed ctrl-D to quit the interactive session.

Also Note: you're not just limited to inspecting the code. You can make any valid Ruby (and in this case, RubyCocoa) calls. Look at RubyInject's README file for more details.

Conclusion

Overall, I am impressed with both RubyCocoa and RubyOSA. Yes, the technology needs to mature a bit. It still has too many rough edges and sharp corners, and you might have trouble finding documentation. Also, RubyOSA still seems harder than it really needs to be. But, all in all, it's nice to have this much power and flexibility built into Leopard.

For the purpose of this Tutorial, I tried to push as much code into Ruby as possible. That, however, may not be the best solution. Ruby and Objective-C have different strengths. This opens up interesting possibilities for hybrid applications. At the very least, you will want to move the bottlenecks from Ruby to Objective-C.

But, there are more interesting opportunities. For example, Ruby has proven quite useful for implementing embedded domain specific languages (DSL). If you don't know what I'm talking about, check out Jim Freeze's excellent introduction at http://www.artima.com/rubycs/articles/ruby_as_dsl.html.

Basically, think of DSLs as mini scripting languages that we build for very specific purposes. Ruby's flexible syntax lets you create mini languages that appear very natural, but remains completely valid ruby syntax—that means, you can execute them directly as Ruby code.

I am a big fan of DSLs in general, and Ruby-based DSLs in particular. Now I can add them to my Objective-C projects. That, alone, is worth the price of admission.

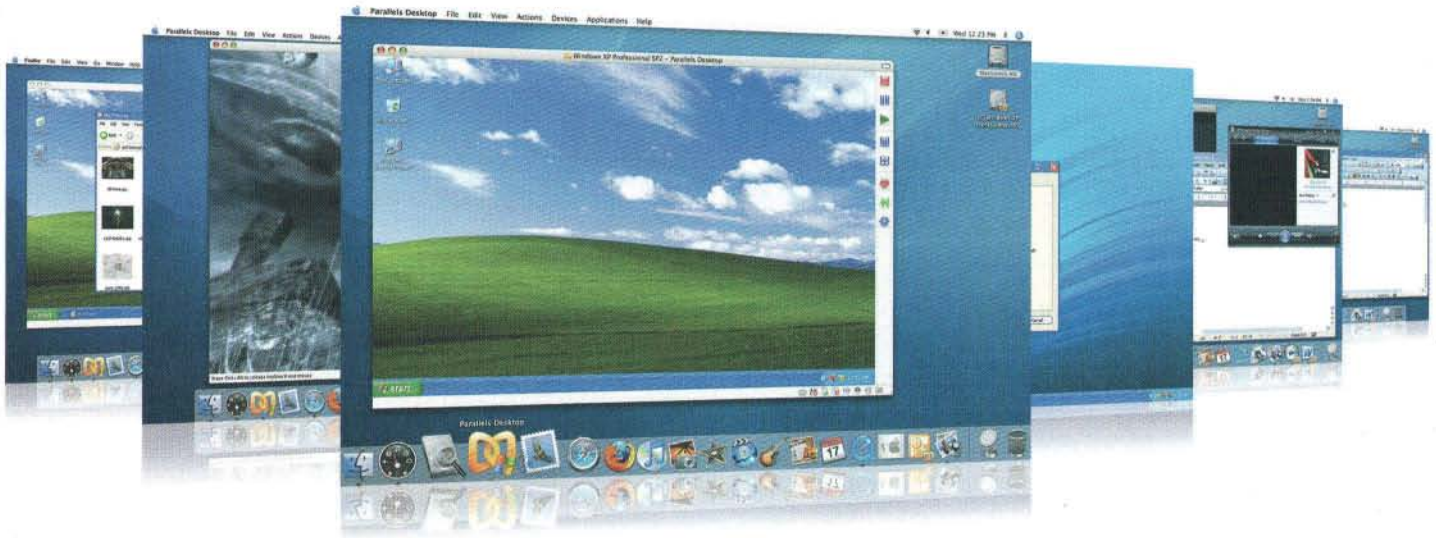
And that's only one example. As time goes on, I'm sure we will find many amazing ways to cross-pollinate Ruby and Objective-C. After all, we haven't even talked about combining a Rails application with Cocoa or OSA. There's bound to be an interesting trick or two there.

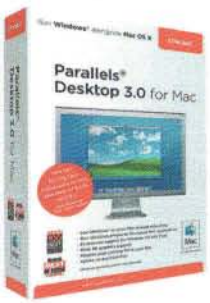
About The Author

Rich Warren lives in Honolulu, Hawaii with his wife, Mika, daughter, Haruko, and his son, Kai. He is a software engineer, freelance writer and part time graduate student. When not playing on the beach, he is probably writing, coding or doing research on his MacBook Pro. You can reach Rich at rikiwarren@mac.com, or check out his blog at http://freelancemadscience.blogspot.com/

Plays well with others.

Run Windows and Mac OS X at the same time - without rebooting.





With Parallels® Desktop 3.0 for Mac, you can:

- Automatically open Windows files with Mac software and Mac files with Windows software
- Bulletproof your virtual machine from Windows system crashes and malware
- Run selected Windows-only 3D games and applications with 3D graphics support
- Browse through Windows files and folders without launching Windows
- Move your existing Windows PC to a Mac without losing data or reinstalling any software
- Completely share files and folders between Windows and Mac

Parallels™

Parallels® Desktop 3.0 for Mac provides even more ground breaking features and capabilities than previous versions. Already The #1 Selling Mac System Utility according to NPD Techworld (09/06 – 12/07) and recently named an InfoWorld Magazine "2008 Technology of the Year", Parallels® Desktop is trusted by over 800,000 users world wide.

Want better Windows and Mac integration? Want to run the hottest PC games and graphics software? Worried about security and system mishaps? Parallels® Desktop 3.0 for Mac includes 50+ new features and enhancements, including a number of integration features and virtual machine utilities unavailable anywhere else.

To discover why Parallels® Desktop is the leading desktop virtualization solution for running Windows on a Mac, visit us online or call us at 1 (425) 282-6405.

PCWORLD 100 BEST PRODUCTS of 2007

Macworld BEST OF SHOW 2007

Download a trial today at www.parallels.com/download

Website – www.parallels.com Email – sales@parallels.com Phone – 1 (425) 282-6405

by Edward Marczak

File Manipulation with PHP

... and other tips

Introduction

We've been talking about PHP as a general scripting language, and how nice it is that it's standard issue on OS X. This makes life as a sysadmin a lot easier when rolling out scripts across many machines, even when they're a mix of, say, 10.3, 10.5, and even Linux and FreeBSD! Topics so far have covered the general language and database access. This month, we'll visit file manipulation and talk about ways to keep growing scripts manageable.

BE-A-U-TI-FUL

PHP has evolved from a bit of a rag-tag just-get-it-done language into a mature, mostly-object-oriented one. Even though version 5 brings stronger OO, you can still use it procedurally. At the very least, you need to know about *functions*, a feature found in most languages.

A *function* is a nearly independent subsection of a larger program. Most importantly, it lets a programmer create a reusable block of code. This, in turn, reduces duplicate code, allows a problem to be broken down into smaller pieces and generally helps code "self-document" and remain manageable. Examples are always best.

To create a function, you start with a *function declaration*, and use curly braces to contain the function's code. Listing 1 contains a very basic sample function.

Listing 1: A sample function

```
function add_em_up($a, $b) {
    $c = $a + $b;
    return $c;
}
```

What is there to immediately learn about this sample? First, a function name follows the same conventions as other labels in PHP. Namely, it starts with a letter or underscore, followed by any number of letters, numbers, or underscores. Following the name, there is a *parameter list*, which can be empty. This function would be called like this:

```
$sum = add_em_up(9,8);
print $sum;
```

Naturally, this will output "17." To pass results back from our function, they must be returned using the `return` keyword. PHP also supports passing by reference, a topic covered later. This raises an issue regarding *scope*, or, "which variable are we talking about, and where is it valid?" Take the code in listing 2.

Listing 2

```
01 $a = 5;
02
03 function set_a($number) {
04 $a = $number;
05 print $a."\n";
06 }
07
08 print $a."\n";
09 set_a(42);
10 print $a."\n";
```

For those who haven't seen this illustrated before, we initially set "$a" to 5. The first print statement – outside of the function – verifies this by printing "5". Line 9 transfers execution to the `set_a` function, starting at line 3. When the function `set_a` is called, "42" is passed in as a *parameter*, and assigned to `$number`. In turn, `$a` is assigned to `$number` – for illustrative purposes only – and then printed. This reveals "42" as expected. Once complete, though, the program is picked up from after the function call, line 10. When the print statement in line 10 prints the value of `$a`, "5" is once again printed. What happened here?

The question is one of *scope*, something Mac in the Shell hasn't dealt with in the past in bash scripting. "Scope" simply refers to where a variable is valid, or, its *context*. Functions create their own *local scope*. Therefore, the "$a" variable defined inside of the "set_a" function is a different variable "$a" than the one used outside of the function. The code in listing 2 should illustrate that quite clearly. What if we *do* want to update variables outside of a function? There are several approaches.

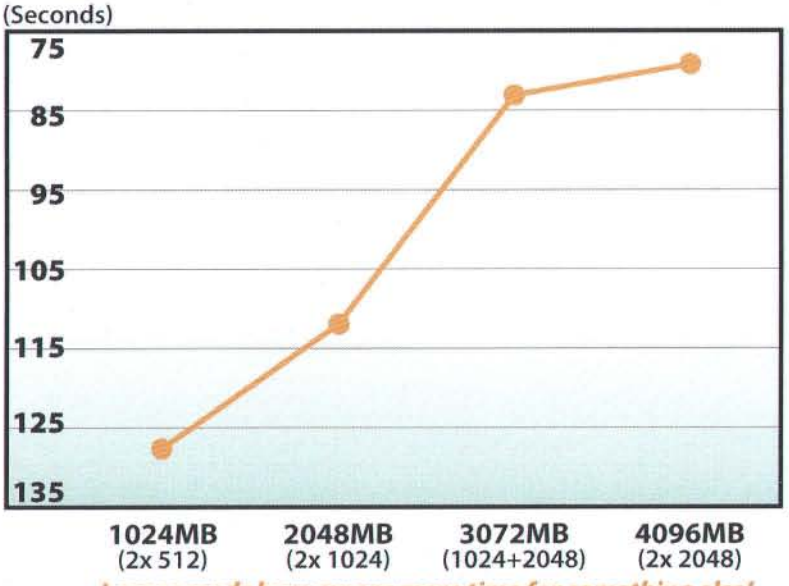
The easiest way to get a value out of a function is to use the `return` keyword, as seen in listing 1. When calling the function, it will be the right hand side of an assignment. Again, to call the function in listing 1, you would use:

```
$sum = add_em_up(9,8);
```

and the returned value would be assigned to the variable `$sum`.

Another way of handling this is via the `global` keyword. By declaring a variable global within a function, all references to that variable will be made to the global version. Listing 3 contains an example.

More Memory = Faster Mac



Adding more memory to your Mac® is fast, easy and inexpensive. And the difference this extra memory makes is AMAZING! Your applications and operating system run faster and more efficiently. With more memory, the only likely time you'll ever see a "spinning beach ball" is at the beach because you've gotten your work done faster!

Visit www.owcmac.com/memory – where you'll find the only Mac focused upgrade guide designed to make it easy to max your Mac.

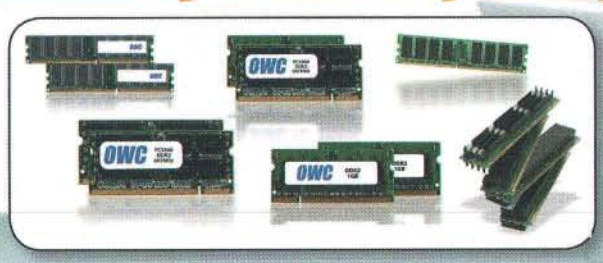
This benchmark measures the time (in seconds) it takes to execute a custom 21-step action script using Adobe® Photoshop® CS3.

MacBook® Pro 17" 2.4GHz Intel® Core™ 2 Duo tested for this benchmark. All benchmark tests use OWC® brand memory, except for the stock memory configuration test. Results vary by machine, but in all test cases, significant gains exist with more installed memory.

1GB as low as **$21.99**

4GB kits from **$89.99**

Memory for nearly every Apple Mac!



All OWC memory meets or exceeds manufacturer specifications for maximum quality assurance, but at the independent prices you can afford. Plus you receive the industry-leading Lifetime Advance Replacement Warranty and award-winning US-based customer support.

Other World Computing
Serving the Mac Universe Since 1988
www.owcmac.com 800.275.4576

BizRate 2007

Listing 3

```
$grand_total = 0;

function compute_totals() {
    global $grand_total;
    $base = read_values();
    $grand_total = $base + ($base * 0.75);
}

compute_totals();
print $grand_total."\n";
```

As you may have guessed in reading listing 3, if `$base` turns out to be 100, the final print statement will actually print "7500" despite $grand_total being referenced outside of the function that assigned it. It's possible to declare as many variables as you like as global to a function. This is one way to manipulate multiple variables from within a function.

Another way to alter the value of several variables external to a function is *passing by reference*. Simply, by default, the value of a variable is passed in to a function, and not the variable itself. What is a variable? It's just a location in memory. Passing by reference actually passes a reference to the memory location of the variable, rather than the value of the variable. An example should make this easier to understand than an explanation. Listing 4 illustrates.

Listing 4

```
$a = 1;
$b = 2;

function swap_them(&$num1, &$num2) {
    $temp=$num1;
    $num1=$num2;
    $num2=$temp;
}

swap_them($a,$b);
print "$a\n";
print "$b\n";
```

In the `swap_them()` function, the variables `$num1` and `$num2` point to the same memory location as the variables you pass in, meaning, altering `$num1` and `$num2` will alter the original variable also. Prepending an ampersand to the argument in the function definition with an ampersand causes that argument to passed by reference. Running the code in listing 4 results in printing:

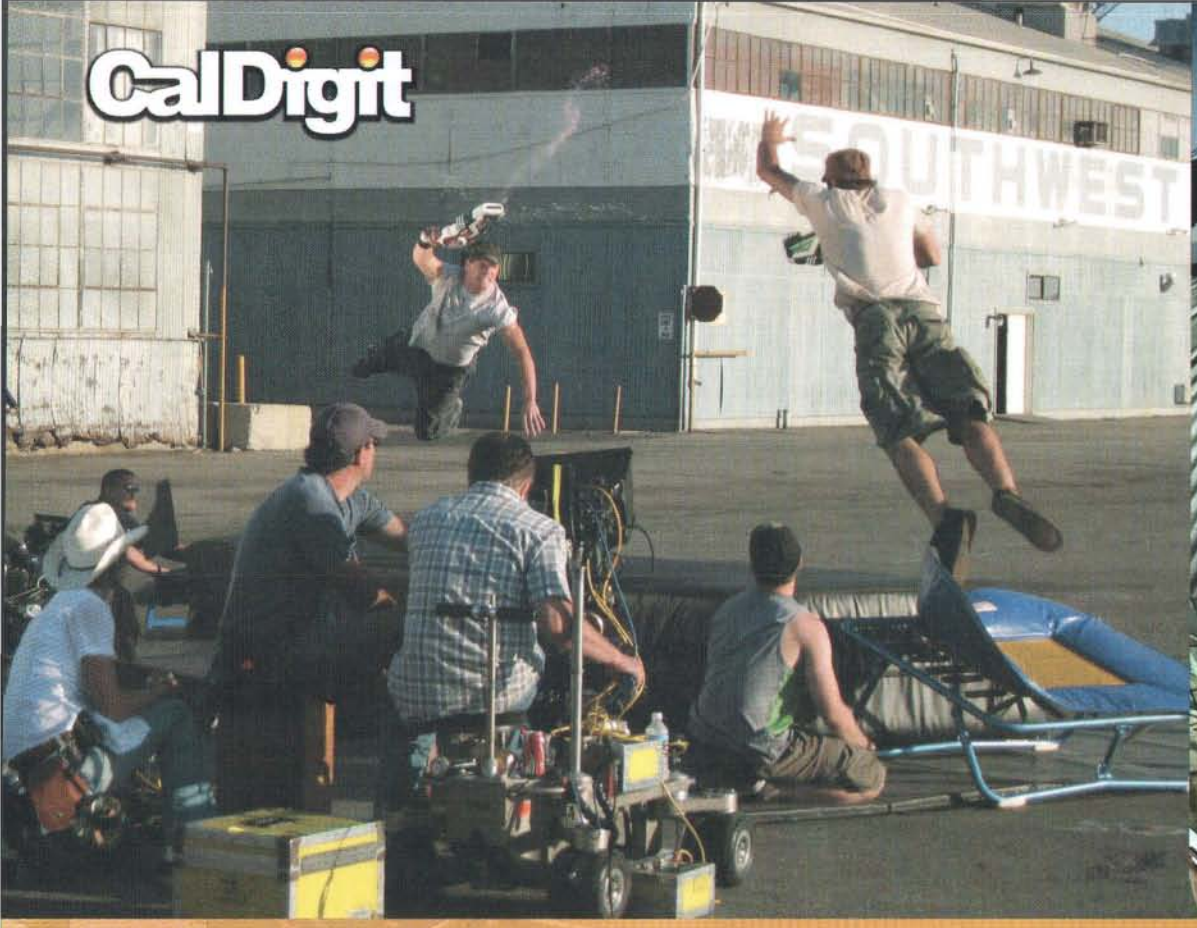
2
1

In other words, the variables get swapped, no returns or globals needed.

Manipulating Files

Now that you know how to create larger, well-formed programs, we can pick up where last month left us hanging. Namely, dealing with files in a more direct way. Last month's sample created a CSV file from data read out of a database.

CalDigit



"I can't afford hiccups when dealing with time sensitive deadlines! CalDigit's HDPro is reliable and delivers. It's the ideal solution for my editorial needs."

-Leo Zahn

HDPro

SCALABLE. SECURE. SCREAMING PERFORMANCE.

- 8 Drives:2TB/2.5TB/4TB/6TB/8TB
- Supports RAID 0,1,5,6,and JBOD

LEO ZAHN - "THE UNCONVENTIONAL PATH"

Los Angeles based Commercial Director/Cameraman/Editor Leo Zahn has directed more than 500 commercials in the United States, Canada and Europe. A visual storyteller with a masterful eye for design, Leo Zahn has found challenges in many arenas: from automotive to food, from electronics to alcoholic beverages, from tabletop to over 200 toy commercials. His work has been rewarded with Clio and Cannes awards.

Most of his commercials require visual effects and multi-layered compositing, well before rough cuts are presented to the client. Having searched for a reliable and cost-effective RAID 5 solution, Leo decided on CalDigit's new HDPRO unit, which delivers screaming read/write rates of more than 380 MB/s. Ever since the HDPRO was installed in Leo's edit bay, it's been working non-stop. It's now the hub for all of his digital assets, scaled from uncompressed telecine footage to HD and 2K.

 http://www.caldigit.com

However, the solution was half-baked: spit the data to standard out and let the user redirect in the shell. Not terrible, but we can certainly be nicer to the user.

PHP has built-in functions that allow file access and manipulation. Let's look at the basics first.

Before any access, a file must be opened using the `fopen` function. `fopen` takes at minimum two parameters: the file and the mode. The file parameter is interesting as we're in an environment where everything is a file. So, the file that you pass in can be a simple on-disk filename like "accounts.txt". It can also be a file descriptor like "php://stdout". Perhaps most interesting is the use of a URI, such as "ftp://public.example.com" or "http://www.example.com". We will only cover the first two types in this article.

The second parameter is the open *mode*. The mode parameter specifies the type of access the program needs to the resource. The following table summarizes the modes available:

mode	Description
'r'	Open for reading only; place the file pointer at the beginning of the file.
'r+'	Open for reading and writing; place the file pointer at the beginning of the file.
'w'	Open for writing only; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
'w+'	Open for reading and writing; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
'a'	Open for writing only; place the file pointer at the end of the file. If the file does not exist, attempt to create it.
'a+'	Open for reading and writing; place the file pointer at the end of the file. If the file does not exist, attempt to create it.
'x'	Create and open for writing only; place the file pointer at the beginning of the file. If the file already exists, the fopen() call will fail by returning **FALSE** and generating an error. If the file does not exist, attempt to create it.
'x+'	Create and open for reading and writing; place the file pointer at the beginning of the file. If the file already exists, the fopen() call will fail by returning **FALSE** and generating an error. If the file does not exist, attempt to create it.

Any of the modes listed may also be supplanted with a *'b'* as the last character of the *mode* parameter. This designates binary mode, and no text or line-ending translations will take place. It is highly recommended that this mode always be used.

After a successful `fopen`, a file can be read using `fgets()`, `fread()` or `file()`. Each must be passed the

Cross-platform that really works.

Create your own software. Easily.

www.realbasic2008.com
Try REALbasic risk-free!
REALbasic includes a 90-day money back guarantee.

REALbasic is the powerful, easy-to-use tool for creating your own software. At REAL Software, we call it a problem solver. You've probably said, "Wouldn't it be great if there was a little application that ..." REALbasic fills that blank.

REALbasic compiles native applications for Mac OS X, Windows and Linux from one set of source code. Each version of your software looks and works just as it should in each environment. You can even create a Universal Binary with one mouse-click.

www.realbasic2008.com

valid file handle resource obtained from `fopen`. A file open for writing can be written to using the `fwrite()` function, which also must be passed a valid handle created by an `fopen()` call. Seeing this all put together will serve us best.

First, create a dummy configuration file and save it in top level of your home directory, named ".ftest.rc" (note the leading dot). Make its contents as follows:

```
daemon=1
send_mail=1
verbose=0
```

Listing 5 will demonstrate both read and write functions.

Listing 5

```
01 <?php
02 $handle = @fopen(getenv("HOME")."/.ftest.rc", "r");
03 if ($handle) {
04  while (!feof($handle)) {
05      $buffer = fgets($handle, 4096);
06          $params = explode("=",$buffer);
07          $flags[trim($params[0])]=trim($params[1]);
08  }
09  fclose($handle);
10  print_r($flags);
11 } else {
12  $stderr = fopen('php://stderr', 'w');
13  fwrite($stderr,"There was an Error\n");
14  fclose($stderr);
15 }
16 ?>
```

Three new things are introduced on line 2: the `fopen()` call, the at-sign ("@") and the `getenv()` call. The `fopen()` call is as discussed – pass in a file and mode. The at-sign in PHP is used to suppress error messages from printing. We can do that in this case as we're testing for success on the next line. Finally, `getenv()` fetches an environment variable.

Line 4 starts a while loop that runs until end of file is detected with the `feof()` function. Then, on line 5, one full line – up to 4,096 characters – are read from the file into the variable `$buffer` using the `fgets()` call.

Line 9 closes the file and line 10 prints out the entire array. Since we're not redirecting this output in any way, PHP will just print as usual: to stdout.

Line 12 begins the `else` sequence - if something is wrong with opening the config file. The variable `$stderr` is a new handle that references stderr. fwrite is used to send the message via stderr, followed by a call to close that resource.

This gives you an example of how to handle file input and output, and how to read or write to any of the standard file descriptors: stdin, stdout and stderr. If the code in listing 5 is run from the command line as such:

```
php ftest.php > flags.txt
```

The output from the print_r statement will be redirected into the "flags.txt" file. The error statement, however, if invoked, will still print to the console, as we haven't redirected stderr.

Summary

In total, we've learned the foundations of PHP, flow control, database access, functions and file manipulation. This alone allows a lot of flexibility and incredible utility. The database access alone makes it a better fit for certain situations over bash scripting. Next month will wrap up this short series on PHP as a general scripting language. We'll touch on some of the small, but significant pieces we've left out thus far, and ways to reach into OS X.

Media of the month: don't let me hold you up...you should probably be reading all of the iPhone SDK documentation you can get your hands on at this point!

Next month is WWDC, where I treasure the opportunity to meet everyone in the community face-to-face. Until then, keep up the scripting!

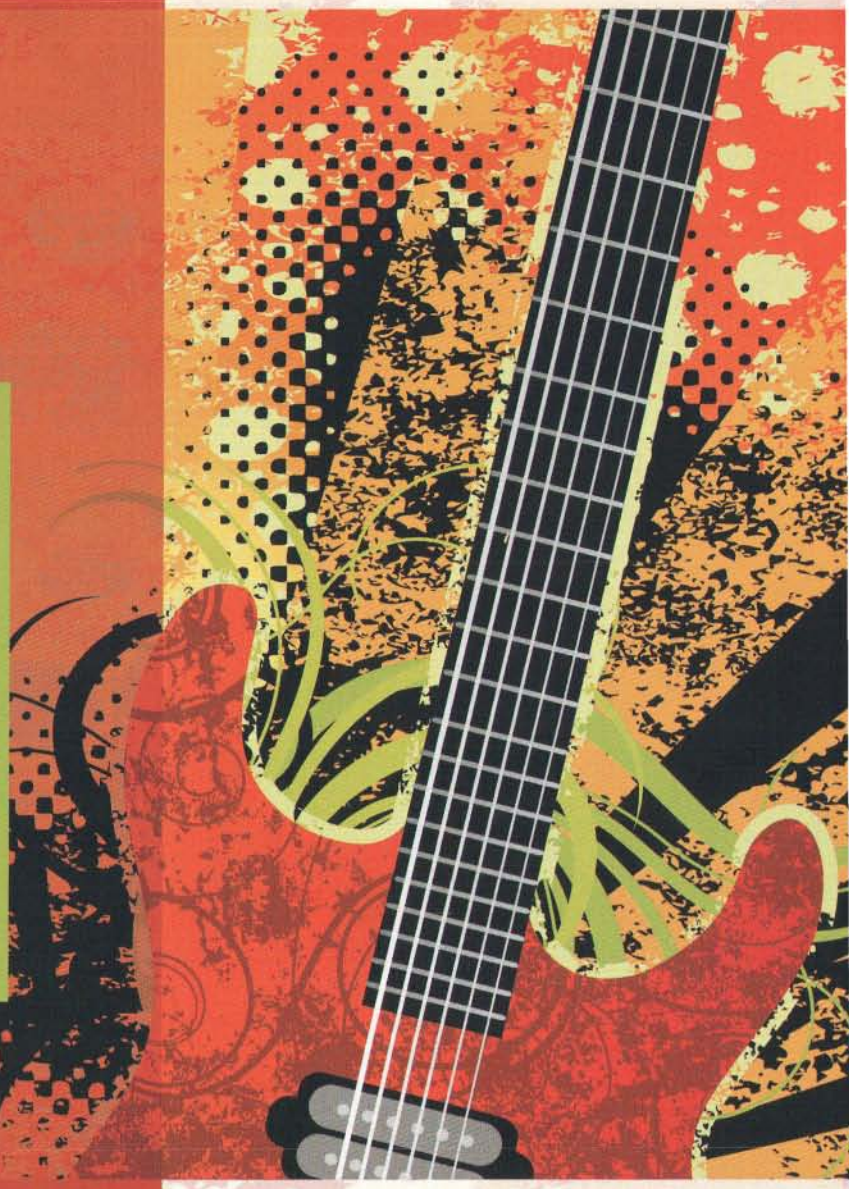
About The Author

Ed Marczak lives and works in New York, along with his wife and two daughters – all of whom have been super understanding while he has been holed up, writing Apple's Advanced System Administration ACSA reference guide. Ed has followed the Macintosh since its inception, and tracked Apple through its various ups and downs since.

LOEWS VANDERBILT HOTEL
NASHVILLE, TN
JUNE 10–13, 2008

VOICES THAT MATTER CONFERENCE

WEB DESIGN



There's no better way to get inspired and unleash your creativity than to join New Riders at the **VOICES THAT MATTER: WEB DESIGN CONFERENCE!** The same editorial team that brings you the books that matter now brings you face-to-face with its esteemed authors and some of the most respected professionals in the industry, all together in one location. Don't miss this opportunity to join the conversation!

Presented by New Riders

Get a FREE Voices That Matter ebook when you sign up to receive more information today at www.voicesthatmatter.com/webdesign08

Media Sponsors

AdobePress · WOW (WWW.WEBPROFESSIONALS.ORG) · DIGITAL WEB MAGAZINE · DesignTools!

The Road To Certification: Part 5: **ACSA and ACTC**

Increase your knowledge and build credibility on the way

by Doug Hanley

Introduction

In this series of articles, we have looked at Apple's IT certifications and hardware certifications. We examined reasons for, and benefits of, getting certified, as well as the testing experience and the changes Apple made to its IT certifications with the release of Mac OS X Leopard, and their new Macintosh Technician certification, which qualifies a technician to perform warranty repairs while working at Apple Authorized Service Providers. We also looked at Apple's Pro Apps certifications for applications like Final Cut Pro and Logic Pro. In this final article in the series, we will review the requirements for the new Apple Certified System Administrator (ACSA) 10.5 certification. We look at an overview of the topics covered on each of the exams required for ACSA certification, and what resources are available to help you prepare. Those resources of course will include Apple Authorized Training Center classes and books. We will also discuss how to become an Apple Certified Trainer for IT courses.

Apple Certified System Administrator

Apple Certified System Administrator (ACSA) is Apple's highest IT certification. An ACSA is someone who has an in-depth knowledge of Mac OS X's technical architecture and can design and maintain networks. They should be able to enable, customize, tune and integrate Mac OS X, Mac OS X Server and other Apple technologies within a multi-platform environment. ACSA certification has undergone a few changes over the years. When the certification first launched for Mac OS X 10.2, it required two exams that were based on concepts covered in two five-day classes – one focused on client and one on server. With Tiger (Mac OS X 10.4) it became a credit-based system with a minimum of 7 current credits required to be an ACSA. Now with Leopard, there has been a change that I believe is for the better. To achieve ACSA on Mac OS X 10.5 you now need to pass four tests: Server Essentials, Directory Services, Deployment, and Advanced System Administration.

Preparing for the Exams

The best way to prepare for any of the ACSA exams is to take the associated class at an Apple Authorized Training Center (AATC). You can find the nearest AATC at: http://training.apple.com/locations. You could also prepare by reviewing the Apple Training Series book published by Peachpit for the particular course, but you will be more thoroughly prepared by participating in a class. The classes are a combination of lectures and hands-on exercises designed to reinforce the concepts covered in the course.

Directory Services

In January's issue, we discussed in detail what is involved in the Server Essentials course. Now we will look at the other three areas of study: Directory Services, Deployment, and Advanced System Administration.

In the four-day Directory Services course, you will learn how to effectively configure Mac OS X computers to access directory services, and how to configure Mac OS X Server to provide directory services in a mixed-platform environment. The course itself will focus on both Mac OS X as a directory service client, and Mac OS X Server as a directory server. Cross platform solutions will be emphasized in both instances. Students using Mac OS X will learn how to use network accounts and Kerberos authentication with any common directory service, such as Apple's Open Directory, Microsoft's

The only true graphical terminal server for Mac OS X

Aqua Connect Terminal Server is the only scalable enterprise grade solution that allows the Mac OS X platform to be deployed to multiple diverse devices, including Macs, PCs, and thin clients. Aqua Connect offers a feature set that truly simplifies OS X administration.

Mac

Universal

Aqua Connect

www.aquaconnect.net
(866) 543-AQUA (2782)

Active Directory, or an industry-standard LDAP server. In working with Mac OS X Server, students will learn how to run a robust, scalable directory system using Apple's Open Directory service. Students will also learn how to use Mac OS X Server to augment an existing directory service infrastructure.

Having a solid understanding of directory services and how they are implemented and integrated in Mac OS X is crucial in deploying multiple servers offering varying services all tied to the same directory structure. You will want this deeper understanding if you want to move beyond single stand-alone deployments of Mac OS X Server.

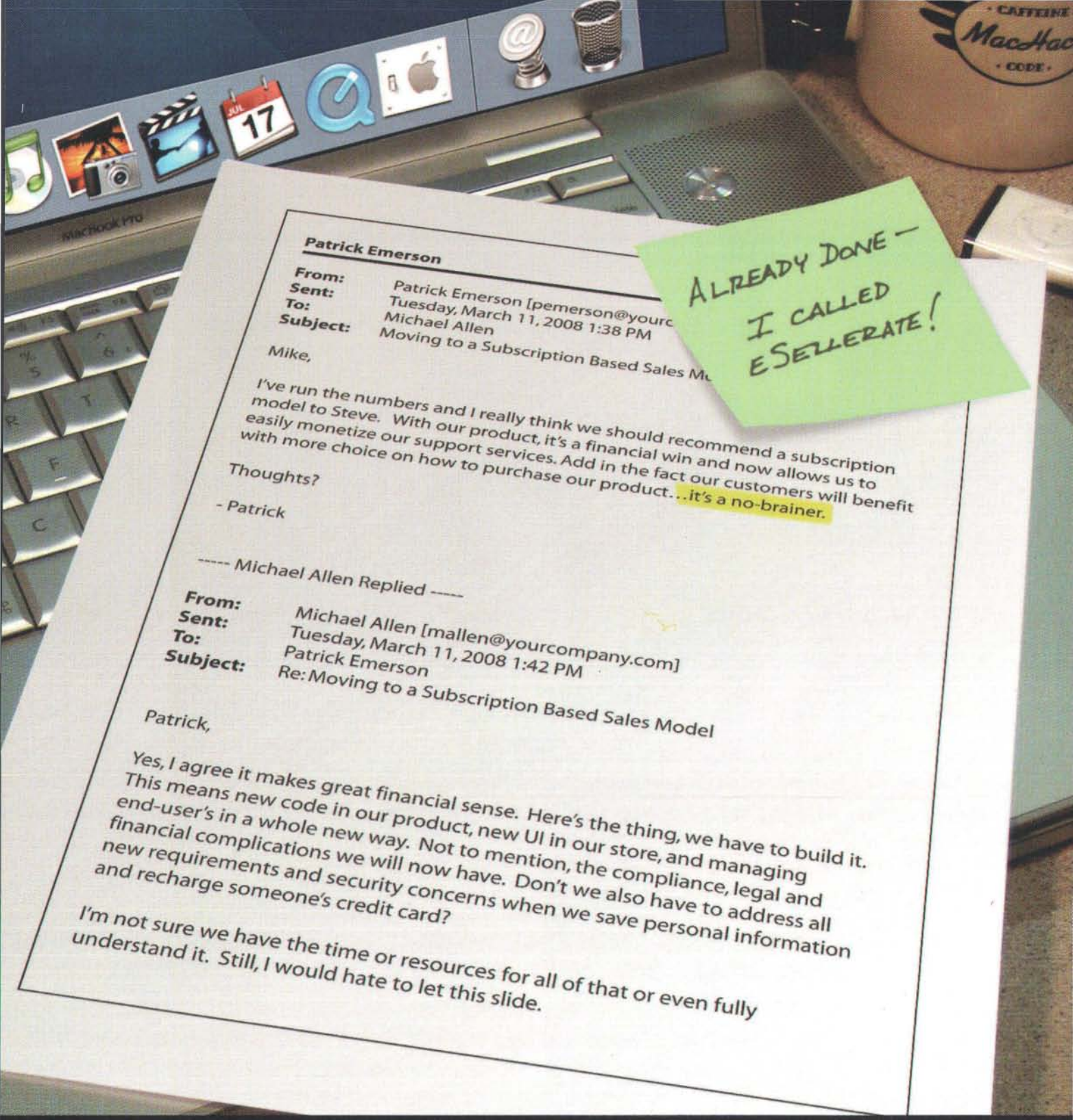
Deployment

Speaking of Deployment, there has to be a better way to install the operating system and software on multiple machines on your network than using a DVD or CD, right? Well that is what is covered in the three-day Mac OS X Deployment course. The first section of Deployment focuses on solutions for deploying software, ranging from individual files to complete system images to multiple machines. Students in this course will get hands-on experience use tools such as Apple Remote Desktop, Disk Utility, PackageMaker, and System Image Utility. You will learn the pros and cons of various deployment solutions. In the second section of the course, students will apply what they have just learned to create a full deployment plan that includes testing, deployment, auditing, and maintenance. You will also learn to create a multi-tiered Software Update Server and third party solutions will be discussed to augment your deployment plan.

Advanced System Administration

The Mac OS X Advanced System Administration course will build on the foundations established in the Support Essentials and Server Essentials courses and is designed to empower students to meet the challenges faced by administrators deploying Mac OS X Server in today's complex and dynamic data centers. This challenging course will be five days in length and provide in-depth and practical skills in Mac OS X technology. The course's task-based focus enhances the learning process through the use of practical examples in a relevant context.

These tasks are organized into several key knowledge domains: implementation, networking, administration, and optimization. Implementation tasks focus on those aspects of installing, upgrading, configuring, and migrating existing legacy systems to more recent versions and configurations. Networking tasks concentrate on establishing solid foundations for network services, as well as connecting private and public networks securely. Students will gain experience with monitoring tools and automation technologies that form the core of the administration tasks necessary to effectively administer larger deployments on a daily basis. An exploration of tools and techniques relating to such performance-based tasks as optimizing services, scaling systems, and establishing high availability of services, data, and components, will help build students' confidence in their administration skills. The course concludes with the vital maintenance tasks that address those aspects of maintaining a system's availability and preserving the integrity of critical data. Extensive use of the command line interface reveals a deeper scope of the course's subject material and prepares students to



become more effective and efficient by taking advantage of the wide variety of automation technologies built into Mac OS X and Mac OS X Server.

When Can I Start?

These last three courses and their accompanying books are being developed at this time. They are expected to be available with their tests by mid 2008. Kevin White, who wrote Support Essentials 10.5 is writing the Peachpit book for Deployment. Arek Dreyer is writing the Directory Services book, and Mac Tech Magazine's own Ed Marczak is feverishly writing the Advance System Administration book. More information about the ACSA certification, classes and other preparatory materials is available on Apple's training website:

http://training.apple.com/certification/acsa

Again, I strongly recommend taking the course at an AATC where you not only get the hands-on experience, access to the books, but also the expertise of a trainer and other peers taking the course. The best trainers are the ones who actually do this type of integration in real world. As you look for a trainer and training center this would be good criteria in choosing your training experience.

Apple Certified Trainer

Are you interested in becoming certified to deliver Apple's curriculum? Well to become an IT Apple Certified Trainer (ACT), you must hold Apple Certified Technical Coordinator (ACTC) certification, v10.4 or higher, have two years of teaching or training experience, and be sponsored by an AATC. That means the AATC vouches for your experience and is willing to use you as a trainer to deliver the course for which you want to become certified. As part of the ACT application, you will be asked to provide references for your teaching/training experience.

You need to obtain ACT certification for each course you wish to teach. To do so, you need to attend the course as a student, pass the student exam at or above the trainer cut score, and submit an ACT application for each course, indicating your sponsoring AATC. You will also have to participate in the course T3 (could be an instructor-led T3 class or one or more podcasts, depending on the course), and pass the trainer-level evaluation (quiz and interview) at the conclusion of the T3 class. When you complete all the requirements, you will receive an email from the ACT Program Manager with instructions for accessing the tools and information available to ACTs.

You can find more information on the ACT program as well as an application at: http://training.apple.com/act

This would be an excellent way to share the knowledge you have gained along the way on your road to certification, by helping others as well.

The End Of The Road?

So as we draw to a conclusion to this series of articles on Apple's certifications we cannot really say that we have reached the end of the road. As long as there is another version of Mac OS X being developed, as long as there are new features being implemented, there will always be more to learn about Apple's operating system and applications. So in reality even if you achieve each and every certification Apple offers on a version of Mac OS X, there will always be a new version coming just over the horizon to learn and gain certification.

In this series of articles we have looked at Apple's three levels of IT Certification, the Apple Certified Support Professional, the Apple Certified Technical Coordinator, and the Apple Certified System Administrator. We have looked at the Apple Certified Macintosh Technician, who is able to perform warranty repairs. Apple's Pro Certifications are for the various creative applications like Final Cut Pro and Logic Pro. Lastly we have looked at the Pro App and IT trainer certifications as well. Hopefully we have provided you with a more informed path on your road to being a certified professional.

Certification is about having a metric that measures your skills and knowledge. It is about being able to show yourself and others you actually know and can operate at certain level in a technical field. So I give you this challenge to continue strive to better yourself and you skills as part of the journey.

About The Author

Doug Hanley owns MacTEK Consulting & Training, an Apple Authorized Training Center in Las Vegas, NV. His time is divided between teaching classes and wrangling servers. He has been providing support on the Mac since the early 90's. To track him down, go to http://www.mactektraining.com ***or email*** doug@mac-tek.com

BookEndz®

Docking Stations for Apple Computers

DOCKING STATIONS FOR APPLE COMPUTERS

Convert your MacBook Pro®, MacBook®, or PowerBook®
into a desktop in seconds
without misplacing cables or damaging connectors.

- Aluminum Plate helps in cooling of notebook
- Connectors are routed to rear of Dock
- MagSafe® Housing prevents accidental disconnect

17" MacBook® Pro Docking Station

13.3" MacBook® Docking Station

- Also available in Black
- Built in DVI/VGA full size
- 5 USB 2.0 compliant ports
- Gigabit Ethernet RJ45

(4 port powered or unpowered hub, AC/DC adapter included)

15.4" MacBook® Pro Docking Station

15" PowerBook® G4 Docking Station

17" PowerBook® G4 Docking Station

12" PowerBook® Docking Station

Visit our website for latest product announcement www.BookEndzdocks.com

BookEndz®
Manufactured by Olympic Controls
1250 Crispin Drive • Elgin, Illinois 60123 • USA
Phone: 847-742-3566 • Fax: 847-742-5686 • Toll Free: 888-622-1199
E-mail: Sales@BookEndzdocks.com

Test-driven Development Using AppleScript—Part 2

Creating AppleScript applications using ASUnit

by Andy Sylvester

Introduction

In the first part of this series, the concept of test-driven development was introduced, showing how to add functionality to an AppleScript application by writing tests first, then the program logic. The ASUnit testing framework was also demonstrated to show how multiple test scripts could be created to test application logic. In the second part of this series, a complete application will be developed using test-driven techniques and the ASUnit testing framework.

Creating the application

To keep things simple, I am going to develop a number guessing game. We will write tests and use those tests to guide the design of the logic for the application. Here is an initial list of "requirements" for our game:

The application will pick a random number from 1 to 100
Players have to pick a number between 1 and 100
The application will tell the player if their guess is out of range or not an integer
The application will tell the player if the number they pick is higher than the random number, lower than the random number, or the right number.
When the player picks the right number, the application will congratulate the player and tell the player how many guesses it took to get the right answer. After that, the game ends.

We could add more requirements for our game, but this is enough to define a complete application. In the first part of this series, we presented the following steps for test-driven development:

Write some test code
Run the test and see that it fails
Write the source code that implements the feature for the test
Run the test again and see that it passes
Refactor or clean up source code

To begin, we will write several tests, focusing on the number that the player chooses (needs to be between 1 and 100 inclusive). Listing 1 shows the first test script:

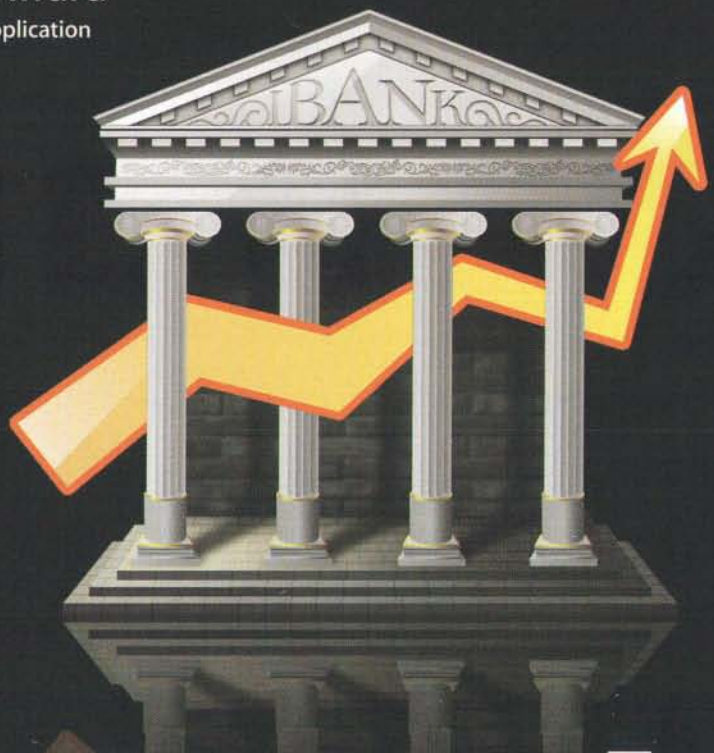
```
property parent : load script file ¬
    (("Sylvester HD:Library:Scripts:") & "ASUnit.scpt")

property suite : makeTestSuite("My Guessing Game Tests")

script |GuessingTests|
    property parent : registerFixture(me)

    on setUp()
    end setUp

    script |GuessOutOfRangeHigh|
        property parent : registerTestCase(me)
        set testGuess to TestGuessingGame's GetNewGuess()
        should(testGuess is greater than 100, ¬
            "testGuess is too high!")
    end script
    script |GuessOutOfRangeLow|
        property parent : registerTestCase(me)
        set testGuess to TestGuessingGame's GetNewGuess()
        should(testGuess is less than 1, ¬
            "testGuess is too low!")
    end script
    script |GuessIsNotAnInteger|
        property parent : registerTestCase(me)
        set testGuess to TestGuessingGame's GetNewGuess()
        set checkTestGuess to int(testGuess)
        should(testGuess is equal to checkTestGuess, ¬
            "testGuess is not an integer!")
    end script
end script
run makeTextTestRunner(suite)
```

Running these tests give the following results:

```
My Guessing Game Tests

GuessingTests - GuessOutOfRangeHigh ... ERROR
GuessingTests - GuessOutOfRangeLow ... ERROR
```

Apple Design Award
Best Mac OS X Leopard Application
2007 Runner-Up

You'll never go back to Quicken



iBank™

Your finances never looked so good.

Manage
- Bank Accounts
- Credit Cards
- Investments

Analyze
- Income
- Expenses

Plan
- Budgets
- Forecasts
- Schedule

Now available at these fine retailers:

MICRO CENTER The Ultimate Computer Store

Apple Store

TEKSERVE

amazon.com and you're done.

Office DEPOT

J&R .com 800 221 8180

OfficeMax

Fry's ELECTRONICS

IGG SOFTWARE

```
GuessingTests - GuessIsNotAnInteger ... ERROR

ERRORS
--------------------------------------------------

test: GuessingTests - GuessOutOfRangeHigh
message: The variable TestGuessingGame is not defined. (-
2753)
--------------------------------------------------

test: GuessingTests - GuessOutOfRangeLow
message: The variable TestGuessingGame is not defined. (-
2753)
--------------------------------------------------

test: GuessingTests - GuessIsNotAnInteger
message: The variable TestGuessingGame is not defined. (-
2753)
--------------------------------------------------

Ran 3 tests in 1 seconds.  passed: 0  skips: 0  errors: 3
failures: 0

FAILED
```

This completes the second step (watching the tests fail). Next, we need to create some application logic so that the tests pass. The tests imply that there is a class called `GuessingGame`, and that it has a method, `GetNewGuess`, that returns a number. Since we do not have a user interface yet, we will create another method (`SetNewGuess`) to set the value of the guess . Using the class we developed earlier as a guide, we can create GuessingGame.scpt as follows:

```
script GuessingGame
    - GuessingGame has one property, the newest guess from
the player.
    property newGuess : 0.0
    - Sets the newGuess property to the value passed to it.
    on SetNewGuess(theGuess)
        set newGuess to theGuess
    end SetDay
    on GetNewGuess()
        return newGuess
    end GetNewGuess
end script
```

Now that we have a class, we will modify the test script to reference the `GuessingGame` script, use the `setUp` function to initialize a copy of the class for our tests, and add a property object for testing. Listing 2 shows the updates:

```
property parent : load script file ¬
    (("Sylvester HD:Library:Scripts:") & "ASUnit.scpt")
property lib : load script file ¬
    (("Sylvester HD:Test:") & "GuessingGame.scpt")

property suite : makeTestSuite("My Guessing Game Tests")

script |GuessingTests|
    property parent : registerFixture(me)

    property TestGuessingGame : missing value

    on setUp()
        copy lib's GuessingGame to TestGuessingGame
        tell TestGuessingGame to SetNewGuess(101)
    end setUp

    script |GuessOutOfRangeHigh|
        property parent : registerTestCase(me)
        set testGuess to TestGuessingGame's GetNewGuess()
        should(testGuess is greater than 100, ¬
            "testGuess is too high!")
    end script
    script |GuessOutOfRangeLow|
        property parent : registerTestCase(me)
        set testGuess to TestGuessingGame's GetNewGuess()
        should(testGuess is less than 1, ¬
            "testGuess is too low!")
    end script
    script |GuessIsNotAnInteger|
        property parent : registerTestCase(me)
        set testGuess to TestGuessingGame's GetNewGuess()
        set checkTestGuess to int(testGuess)
        should(testGuess is equal to checkTestGuess, ¬
            "testGuess is not an integer!")
    end script
end script
run makeTextTestRunner(suite)
```

Let's see how our tests are doing now:

```
My Guessing Game Tests

GuessingTests - GuessOutOfRangeHigh ... ok
GuessingTests - GuessOutOfRangeLow ... FAIL
GuessingTests - GuessIsNotAnInteger ... ERROR

ERRORS
--------------------------------------------------

test: GuessingTests - GuessIsNotAnInteger
message: Çscript |GuessIsNotAnInteger|È doesn't understand
the int message. (-1708)
--------------------------------------------------

FAILURES
--------------------------------------------------

test: GuessingTests - GuessOutOfRangeLow
message: testGuess is too low!
--------------------------------------------------

Ran 3 tests in 0 seconds.  passed: 1  skips: 0  errors: 1
failures: 1

FAILED
```

The `GuessOutOfRangeHigh` test passed, but we initialized the value of the guess to 101, so that test should have failed. The `GuessOutOfRangeLow` failed, which shouldn't happen, since the number was greater than 1 (it should have passed!). Finally, the `GuessIsNotAnInteger` had an error (`script |GuessIsNotAnInteger|È doesn't understand the int message`). There must be some problem in how the "int" function is being used. Let's use these results to improve our tests and maybe our code as well.

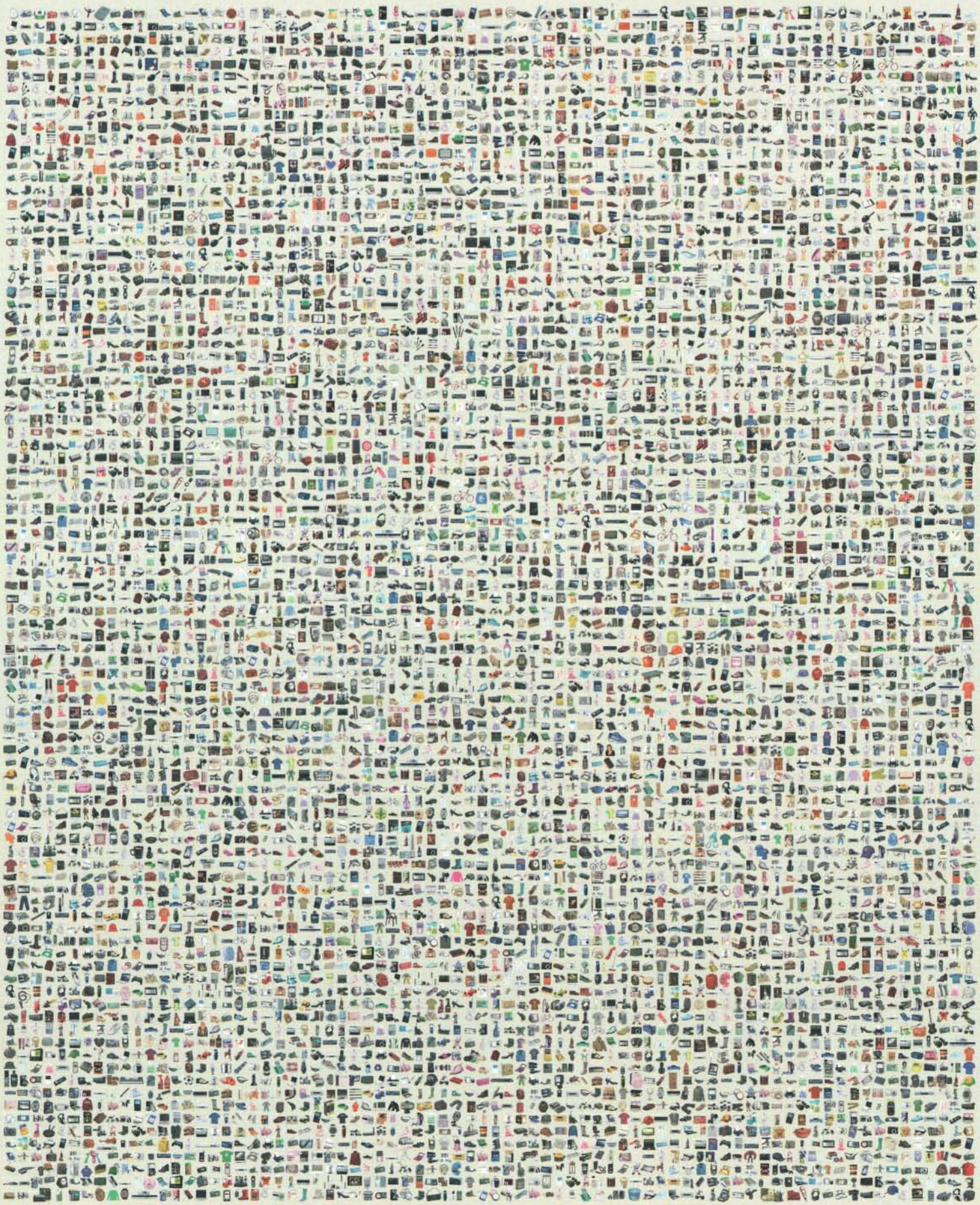
When using the `should` statement in a test, the test fails if the first argument or condition is not true. In the `GuessOutOfRangeHigh` test, the condition is true (101 is greater than 100), but the title of the test suggests it should fail if `testGuess` is greater than 100. Let's try changing the condition to read "`testGuess is less than 101`", so that if `testGuess` is greater than 100, the test will fail. Looking at the `GuessOutOfRangeLow`, the condtion is false, but the number is actually not too low. As in the `GuessOutOfRangeHigh` test, we should change the test condition to match up better with the title of the test. Let's try changing the condition to read "`testGuess is greater than 0`", so that if the value of `testGuess` is less than 1, the test will fail. We will leave the last test as it is and try to fix it in the next pass. Listing 3 shows the updates:

THINGS MADE IN CHINA

COMPRESSED BY

STUFFIT DELUXE

SOFTWARE FOR MAC/PC



```
property parent : load script file ¬
    (("Sylvester HD:Library:Scripts:") & "ASUnit.scpt")
property lib : load script file (("Sylvester HD:Test:") &
"GuessingGame.scpt")

property suite : makeTestSuite("My Guessing Game Tests")

script |GuessingTests|
    property parent : registerFixture(me)

    property TestGuessingGame : missing value

    on setUp()
        copy lib's GuessingGame to TestGuessingGame
        tell TestGuessingGame to SetNewGuess(101)
    end setUp

    script |GuessOutOfRangeHigh|
        property parent : registerTestCase(me)
        set testGuess to TestGuessingGame's GetNewGuess()
        should(testGuess is less than 101, ¬
            "testGuess is too high!")
    end script
    script |GuessOutOfRangeLow|
        property parent : registerTestCase(me)
        set testGuess to TestGuessingGame's GetNewGuess()
        should(testGuess is greater than 0, ¬
            "testGuess is too low!")
    end script
    script |GuessIsNotAnInteger|
        property parent : registerTestCase(me)
        set testGuess to TestGuessingGame's GetNewGuess()
        set checkTestGuess to int(testGuess)
        should(testGuess is equal to checkTestGuess, ¬
            "testGuess is not an integer!")
    end script
end script

run makeTextTestRunner(suite)
```

When we run the tests again, we get the following results:

```
My Guessing Game Tests

GuessingTests - GuessOutOfRangeHigh ... FAIL
GuessingTests - GuessOutOfRangeLow ... ok
GuessingTests - GuessIsNotAnInteger ... ERROR

ERRORS
----------------------------------------------------------------------
test: GuessingTests - GuessIsNotAnInteger
message: Çscript |GuessIsNotAnInteger|È doesn't understand
the int message. (-1708)
----------------------------------------------------------------------

FAILURES
----------------------------------------------------------------------
test: GuessingTests - GuessOutOfRangeHigh
message: testGuess is too high!
----------------------------------------------------------------------

Ran 3 tests in 0 seconds.  passed: 1  skips: 0  errors: 1
failures: 1

FAILED
```

Now the first two tests respond as we expect. The value of `testGuess` is too high, so that test should fail. It is not too low, so that test passed. Now, let's see if we can figure out what the problem is with the third test. In AppleScript, the `as` keyword is used for coercion of a parameter to another type. We will replace the coercion line in the `GuessIsNotAnInteger` test with the following:

```
        set checkTestGuess to testGuess
        checkTestGuess as integer
```

With these changes, we get the following results:

```
My Guessing Game Tests

GuessingTests - GuessOutOfRangeHigh ... FAIL
GuessingTests - GuessOutOfRangeLow ... ok
GuessingTests - GuessIsNotAnInteger ... ok

FAILURES
----------------------------------------------------------------------
test: GuessingTests - GuessOutOfRangeHigh
message: testGuess is too high!
----------------------------------------------------------------------

Ran 3 tests in 1 seconds.  passed: 2  skips: 0  errors: 0
failures: 1

FAILED
```

Now that the tests are working as expected, let's flesh out the logic in our class functions. To start, we will create a function to check the value of the number and create a text response. Listing 4 shows the updates to GuessingGame.scpt:

```
script GuessingGame
    property newGuess : 0.0
    property checkTestGuess : 0
    property guessResponse : "test"
    property rangeResponse : "test"
    -- Sets the newGuess property to the value passed to it.
    on SetNewGuess(theGuess)
        set newGuess to theGuess
    end SetNewGuess
    on GetNewGuess()
        set guessResponse to CheckGuessCases()
        return guessResponse
    end GetNewGuess
    on CheckGuessCases()
        if newGuess is greater than 100 then
            set rangeResponse to ¬
            "Your guess is too high! Pick a number from 1 to
100."
        end if
        if newGuess is less than 1 then
            set rangeResponse to ¬
            "Your guess is too low! Pick a number from 1 to
100."
        end if
        set checkTestGuess to newGuess
        checkTestGuess as integer
        if checkTestGuess is not equal to newGuess then
            set rangeResponse to ¬
             "Your guess is not an integer! Pick a number
from 1 to 100."
        end if
        return rangeResponse
    end CheckGuessCases
end script
```

After making updates to GuessingGame.scpt, you will need to quit Script Editor and start it again for the changes to take effect. Running the tests give the following results:

```
My Guessing Game Tests
```

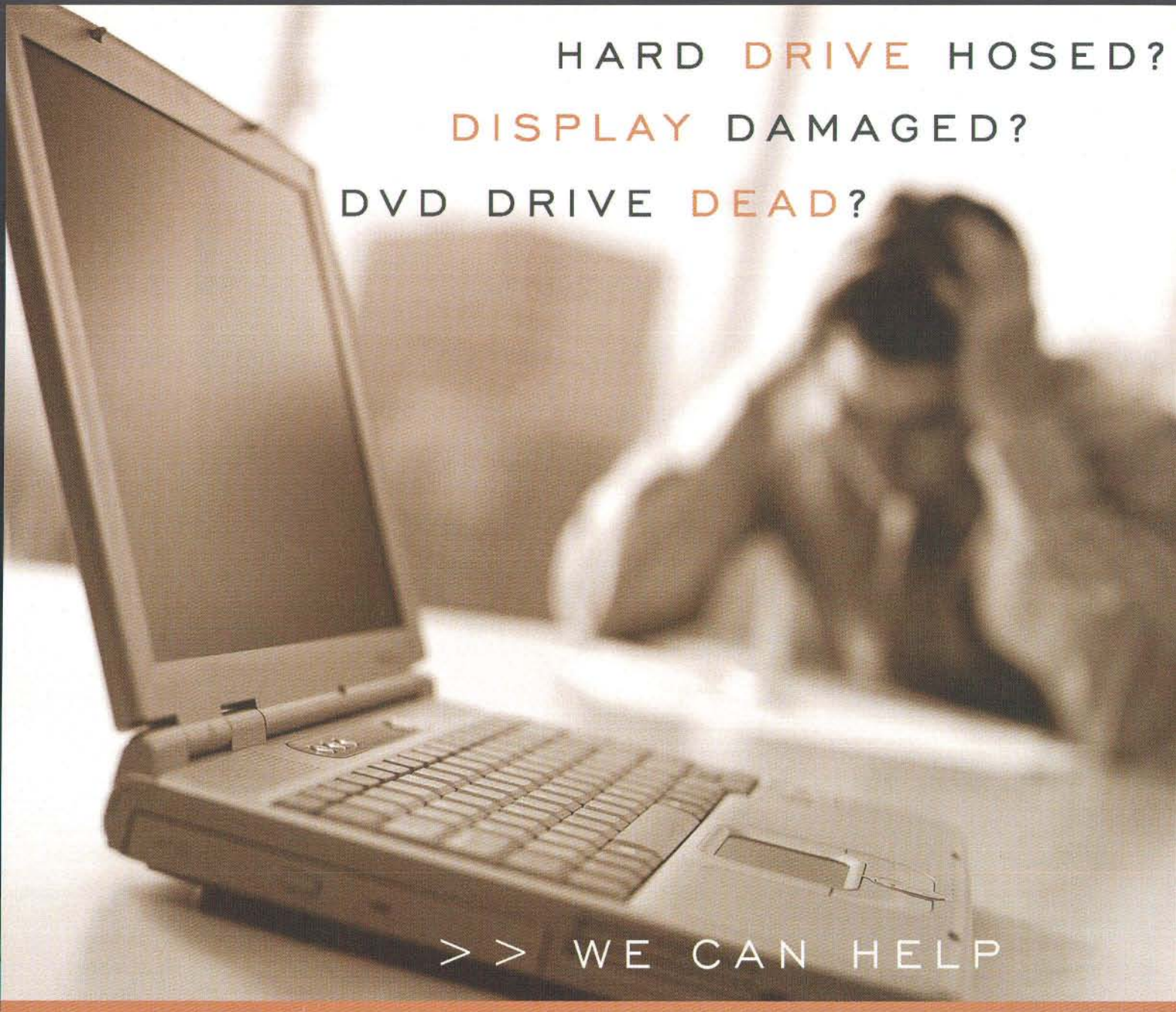
```
GuessingTests - GuessOutOfRangeHigh ... FAIL
GuessingTests - GuessOutOfRangeLow ... ok
GuessingTests - GuessIsNotAnInteger ... ERROR

ERRORS
--------------------------------------------------------------------

test: GuessingTests - GuessIsNotAnInteger
message: Can't make "Your guess is too high! Pick a number
from 1 to 100." into type integer. (-1700)
--------------------------------------------------------------------

FAILURES
--------------------------------------------------------------------

test: GuessingTests - GuessOutOfRangeHigh
message: testGuess is too high!
--------------------------------------------------------------------

Ran 3 tests in 0 seconds.  passed: 1  skips: 0  errors: 1
failures: 1

FAILED
```

Now that the class functions return a text response, we need to update the tests to check for that response. Actually, we want the tests to check to make sure that we did not get any of the error responses from `GetNewGuess`. If we get one of those responses, we know we will have a test failure. Listing 5 shows the test updates:

```
property parent : load script file ¬
    (("Sylvester HD:Library:Scripts:") & "ASUnit.scpt")
property lib : load script file (("Sylvester HD:Test:") &
"GuessingGame.scpt")

property suite : makeTestSuite("My Guessing Game Tests")

script |GuessingTests|
    property parent : registerFixture(me)

    property TestGuessingGame : missing value

    on setUp()
        copy lib's GuessingGame to TestGuessingGame
        tell TestGuessingGame to SetNewGuess(101)
    end setUp

    script |GuessOutOfRangeHigh|
        property parent : registerTestCase(me)
        set testGuess to TestGuessingGame's GetNewGuess()
        should(testGuess to ¬
            "Your guess is too high! Pick a number from 1 to
100.",¬
            "testGuess is too high!")
    end script
    script |GuessOutOfRangeLow|
        property parent : registerTestCase(me)
        set testGuess to TestGuessingGame's GetNewGuess()
        should(testGuess is not equal to ¬
            "Your guess is too low! Pick a number from 1 to
100.", ¬
            "testGuess is too low!")
    end script
    script |GuessIsNotAnInteger|
        property parent : registerTestCase(me)
        set testGuess to TestGuessingGame's GetNewGuess()
        should(testGuess is not equal to ¬
            "Your guess is not an integer! Pick a number from
1 to 100.",¬
            "testGuess is not an integer!")
    end script
end script
run makeTextTestRunner(suite)
```

We will now run a set of tests for each failure condition, and a test where the value of `newGuess` is within range. In the `setUp` script, set the test value to 101. Running the tests gives the following results:

```
My Guessing Game Tests
```



● REPLACEMENT PARTS ● UPGRADES ● REPAIRS FOR YOUR MAC LAPTOP

- FREE ONLINE DO-IT-YOURSELF GUIDES
- SEND-IN REPAIR SERVICE
- BATTERY REPLACEMENTS FROM $ 94.95
- POWER ADAPTERS FROM $ 24.95

VISIT US ON THE WEB>>
OR CALL 866.726.3342

PowerbookMedic.com

Your Mac — Our Patient

www.PowerbookMedic.com
555 Sparkman Dr. Ste. 1620
Huntsville, AL 35816
Tel/Fax: 866.726.3342

```
GuessingTests - GuessOutOfRangeHigh ... FAIL
GuessingTests - GuessOutOfRangeLow ... ok
GuessingTests - GuessIsNotAnInteger ... ok

FAILURES
----------
test: GuessingTests - GuessOutOfRangeHigh
message: testGuess is too high!
----------

Ran 3 tests in 1 seconds.  passed: 2  skips: 0  errors: 0
failures: 1

FAILED
```

The GuessOutOfRangeHigh test failed as expected. Now, set the test value to 0. You should then see the following test results:

```
My Guessing Game Tests

GuessingTests - GuessOutOfRangeHigh ... ok
GuessingTests - GuessOutOfRangeLow ... FAIL
GuessingTests - GuessIsNotAnInteger ... ok

FAILURES
----------
test: GuessingTests - GuessOutOfRangeLow
message: testGuess is too low!
----------

Ran 3 tests in 0 seconds.  passed: 2  skips: 0  errors: 0
failures: 1

FAILED
```

Now, set the test value to "test". You should then see the following test results:

```
My Guessing Game Tests

GuessingTests - GuessOutOfRangeHigh ... ERROR
GuessingTests - GuessOutOfRangeLow ... ERROR
GuessingTests - GuessIsNotAnInteger ... ERROR

ERRORS
----------
test: GuessingTests - GuessOutOfRangeHigh
message: Can't make "test" into type integer. (-1700)
----------
test: GuessingTests - GuessOutOfRangeLow
message: Can't make "test" into type integer. (-1700)
----------
test: GuessingTests - GuessIsNotAnInteger
message: Can't make "test" into type integer. (-1700)
----------

Ran 3 tests in 1 seconds.  passed: 0  skips: 0  errors: 3
failures: 0

FAILED
```

Oops! Looks like we have a problem...what should we do? It looks like CheckGuessCases is not adequately checking for the case when the input is not an integer (in this case, a string). We will restructure this function in GuessingGame.scpt to check for the data class of the input, and arrange the branching structure to stop as soon as the correct condition is detected. Also, we will add another branch to set rangeResponse to a separate message if the number is in range.

```
on CheckGuessCases()
    set guessClass to class of newGuess
    if guessClass is not equal to integer then
        set rangeResponse to ¬
        "Your guess is not an integer! Pick a number
from 1 to 100."
    else if newGuess is greater than 100 then
        set rangeResponse to ¬
        "Your guess is too high! Pick a number from 1
to 100."
    else if newGuess is less than 1 then
        set rangeResponse to ¬
        "Your guess is too low! Pick a number from 1
to 100."
    else
        set rangeResponse to "Guess is within range."
    end if
    return rangeResponse
end CheckGuessCases
```

Now, set the test value to "test" again. You should then see the following test results:

```
My Guessing Game Tests

GuessingTests - GuessOutOfRangeHigh ... ok
GuessingTests - GuessOutOfRangeLow ... ok
GuessingTests - GuessIsNotAnInteger ... FAIL

FAILURES
----------
test: GuessingTests - GuessIsNotAnInteger
message: testGuess is not an integer!
----------

Ran 3 tests in 0 seconds.  passed: 2  skips: 0  errors: 0
failures: 1

FAILED
```

This gives the failure response that we expect. Since we had a test problem with the text entry, let's try setting the test value to a real number, like 30.7. You should then get the following results:

```
My Guessing Game Tests

GuessingTests - GuessOutOfRangeHigh ... ok
GuessingTests - GuessOutOfRangeLow ... ok
GuessingTests - GuessIsNotAnInteger ... FAIL

FAILURES
----------
test: GuessingTests - GuessIsNotAnInteger
message: testGuess is not an integer!
----------

Ran 3 tests in 0 seconds.  passed: 2  skips: 0  errors: 0
failures: 1

FAILED
```

Again, this is the expected response. Now for the final test, an integer value between 1 and 100 (let's say 51). Our results are:

```
My Guessing Game Tests

GuessingTests - GuessOutOfRangeHigh ... ok
GuessingTests - GuessOutOfRangeLow ... ok
GuessingTests - GuessIsNotAnInteger ... ok

Ran 3 tests in 0 seconds.  passed: 3  skips: 0  errors: 0
failures: 0
```

```
OK
```

Finally, back to where we want to be - all tests passing with proper inputs, and all tests failing when the conditions for the test failures are met.

Now, let's start work on the user interface by modifying `SetNewGuess` in GuessingGame.scpt to display a dialog box and ask the user for a value:

```
on SetNewGuess(theGuess)
    set newGuess to text returned of ¬
    (display dialog "Pick a number from 1 to 100" ¬
        default answer "")
end SetNewGuess
```

This will prompt the user for an input for each of the three tests we have written. When we run the tests with an input of "8", we get the following results:

```
My Guessing Game Tests

GuessingTests - GuessOutOfRangeHigh ... ok
GuessingTests - GuessOutOfRangeLow ... ok
GuessingTests - GuessIsNotAnInteger ... FAIL

FAILURES

----------------------------------------------------------------------

test: GuessingTests - GuessIsNotAnInteger
message: testGuess is not an integer!

----------------------------------------------------------------------

Ran 3 tests in 7 seconds.  passed: 2  skips: 0  errors: 0
failures: 1

FAILED
```

It looks like the code in `CheckGuessCases` is not detecting that the input is an integer. We will add a line to this script in GuessingGame.scpt to check what the class of our input is:

```
on CheckGuessCases()
    set guessClass to class of newGuess
    display dialog "guessClass is " ¬
        & guessClass -- this is the new line
```

When we repeat the tests, the dialog box showed "`guessClass is Unicode text`". It looks like we need more logic to force the input to be a number and to keep the range tests from failing. Let's try restructuring the logic for `CheckGuessCases` again in GuessingGame.scpt to catch the "not an integer" error:

```
on CheckGuessCases()
    try
        set checkNewGuess to newGuess as number
        if checkNewGuess is greater than 100 then
            set rangeResponse to ¬
            "Your guess is too high! Pick a number
from 1 to 100."
        else if checkNewGuess is less than 1 then
            set rangeResponse to ¬
            "Your guess is too low! Pick a number
from 1 to 100."
        else
            set rangeResponse to "Guess is within
range."
        end if
    on error
        set rangeResponse to ¬
        "Your guess is not an integer! Pick a number
```

```
from 1 to 100."
            end try
            return rangeResponse
        end CheckGuessCases
```

We also need to add a property for the `checkNewGuess` parameter to the GuessingGame script:

```
property checkNewGuess : 0
```

When we repeat the tests using the word "test" as the input in the dialog box, we get the following response:

```
My Guessing Game Tests

GuessingTests - GuessOutOfRangeHigh ... ok
GuessingTests - GuessOutOfRangeLow ... ok
GuessingTests - GuessIsNotAnInteger ... FAIL

FAILURES
_____________________________________

test: GuessingTests - GuessIsNotAnInteger
message: testGuess is not an integer!
_____________________________________

Ran 3 tests in 13 seconds.  passed: 2  skips: 0  errors: 0
failures: 1

FAILED
```

Now the restructured logic catches the non-integer value and allows the other tests to pass. Next, we will add some logic to check to see if the guess by the player is above or below the random value. Following our test-driven development philosophy, we will create several new tests in a new test file. Listing 6 shows the new tests:

```
property parent : load script file ¬
    (("Sylvester HD:Library:Scripts:") & "ASUnit.scpt")
property lib : load script file ¬
    (("Sylvester HD:Test:") & "GuessingGame.scpt")

property suite : makeTestSuite("My Guessing Game Tests")

script |MoreGuessingTests|
    property parent : registerFixture(me)

    property TestGuessingGame : missing value

    on setUp()
        copy lib's GuessingGame to TestGuessingGame
    end setUp

    script |GuessIsHigherThanRandomNumber|
        property parent : registerTestCase(me)
        on setUp()
            copy lib's GuessingGame to TestGuessingGame
            set TestGuessingGame's testNumber to 40
            tell TestGuessingGame to SetNewGuess()
        end setUp
        set testGuess to TestGuessingGame's GetNewGuess()
        set GuessIsHighOrLow to TestGuessingGame's
CheckTheGuess()
        should(GuessIsHighOrLow is not equal to ¬
            "Your guess is too high! Pick a lower number.", ¬
            "guess is too high!")
    end script
    script |GuessIsLowerThanRandomNumber|
        property parent : registerTestCase(me)
```

Revolutionize your Lifecycle Management Strategy with LANrev!

True Cross-Platform Lifecycle Management has arrived!
Do you need to manage PCs with a Mac? Macs with a PC?
LANrev is the first client management solution with true cross platform functionality for today's heterogeneous client environments.

Disk Imaging
Asset Inventory
Software Distribution
License Management
Patch Management
Remote Management
Role-Based Administration
Change and Configuration Management
LANrev TheftTrack Client Recovery

LANrev is a scalable multi-platform solution, which can operate with equal efficiency on a single server deployment in the small to medium enterprise, or in a distributed architecture, managing thousands of desktops in a large enterprise.

LANrev's modern, modular platform allows seamless integration into other desktop management tools such as Microsoft's SMS, enabling your organization to leverage previous investments while maximizing current and future efficiencies through the use of innovative new technology such as LANrev. The best part? You can take advantage of your current hardware infrastructure. Unlike the other guys, LANrev does not have massive, dedicated server requirements and you won't need a team of overpaid consultants to deploy and use LANrev.

See for yourself how you can increase your IT efficiency and performance while lowering your cost by using LANrev in your computing environment. Schedule a free web demo and trial license by calling (214) 459-0136 or visiting www.lanrev.com.

Full Mac OS X 10.5 Leopard Support!

LANrev

Mac Universal for windows

```
        on setUp()
            copy lib's GuessingGame to TestGuessingGame
            set TestGuessingGame's testNumber to 40
            tell TestGuessingGame to SetNewGuess()
        end setUp
        set testGuess to TestGuessingGame's GetNewGuess()
        set GuessIsHighOrLow to TestGuessingGame's
CheckTheGuess()
        should(GuessIsHighOrLow is not equal to ¬
            "Your guess is too low! Pick a higher number.", ¬
            "guess is too low!")
    end script
end script
run makeTextTestRunner(suite)
```

In these two tests, we have created `testNumber` to serve as the "random number", then created a reference to a new function, `CheckTheGuess`, which will return a text string telling the user whether the guess is too high or too low. To make the GuessingGame script consistent, we will delete the calling argument for `SetNewGuess` and add a property for `testNumber`. The updated version of GuessingGame.scpt is shown in Listing 7:

```
script GuessingGame
    property newGuess : 0.0
    property checkNewGuess : 0
    property testNumber : 0
    property guessResponse : "test"
    property rangeResponse : "test"
    - Sets the newGuess property to the value passed to it.
    on SetNewGuess()
        set newGuess to text returned of ¬
        (display dialog "Pick a number from 1 to 100" default
answer "")
    end SetNewGuess
    on GetNewGuess()
        set guessResponse to CheckGuessCases()
        return guessResponse
    end GetNewGuess
    on CheckGuessCases()
        try
            set checkNewGuess to newGuess as number
            if checkNewGuess is greater than 100 then
                set rangeResponse to ¬
                "Your guess is too high! Pick a number from 1
to 100."
            else if checkNewGuess is less than 1 then
                set rangeResponse to ¬
                "Your guess is too low! Pick a number from 1
to 100."
            else
                set rangeResponse to "Guess is within range."
            end if
        on error
            set rangeResponse to ¬
            "Your guess is not an integer! Pick a number from
1 to 100."
        end try
        return rangeResponse
    end CheckGuessCases
end script
```

When we run the tests with an input value of 8 for the user's guess, we get the following results:

```
My Guessing Game Tests

MoreGuessingTests - GuessIsHigherThanRandomNumber ... ERROR
MoreGuessingTests - GuessIsLowerThanRandomNumber ... ERROR

ERRORS
----------------------------------------------------------------------
test: MoreGuessingTests - GuessIsHigherThanRandomNumber
message: ÇscriptÈ doesn't understand the CheckTheGuess
message. (-1708)
----------------------------------------------------------------------
test: MoreGuessingTests - GuessIsLowerThanRandomNumber
message: ÇscriptÈ doesn't understand the CheckTheGuess
message. (-1708)
----------------------------------------------------------------------

Ran 2 tests in 21 seconds.  passed: 0  skips: 0  errors: 2
failures: 0

FAILED
```

Now that we know that the tests fail, we will create the `CheckTheGuess` function and add this to GuessingGame.scpt:

```
    on CheckTheGuess()
        if checkNewGuess is less than testNumber then
            set checkResponse to ¬
            "Your guess is too low! Pick a higher number."
        else if checkNewGuess is greater than testNumber then
            set checkResponse to ¬
            "Your guess is too high! Pick a lower number."
        else
            set checkResponse to "You picked the right
number!"
        end if
        return checkResponse
    end CheckTheGuess
```

We also need to add a property for the `checkResponse` element to GuessingGame.scpt:

```
    property checkResponse : "test"
```

Since we are setting `testNumber` to 40 in each test, we will enter the wrong input for the two new tests to see if we can detect a failure. When we input 60 for the `GuessIsHigherThanRandomNumber` test and 20 for the `GuessIsLowerThanRandomNumber` test, we get the following results:

```
My Guessing Game Tests

MoreGuessingTests - GuessIsHigherThanRandomNumber ... FAIL
MoreGuessingTests - GuessIsLowerThanRandomNumber ... FAIL

FAILURES
----------------------------------------------------------------------
test: MoreGuessingTests - GuessIsHigherThanRandomNumber
message: guess is too high!
----------------------------------------------------------------------
test: MoreGuessingTests - GuessIsLowerThanRandomNumber
message: guess is too low!
----------------------------------------------------------------------

Ran 2 tests in 4 seconds.  passed: 0  skips: 0  errors: 0
failures: 2

FAILED
```

We will now add a test for the condition where the user picks the right number:

```
    script |GuessIsCorrect|
        property parent : registerTestCase(me)
```

Other World Computing™

Serving the Mac Universe since 1988

Excellent service | Competitive prices | Quality products | Expert tech staff

Maximize your data storage.

OWC® Mercury Elite Pro™ Aluminum

All Mercury Elite Pro Aluminum models are ideally configured for Audio, Video, Digital Photography, Professional Music, Graphics, General Data and Back-Up applications.

- Custom high performance Oxford Bridge.
- Rugged, machined aluminum enclosure & super quiet operation.
- Ultra-protective shock isolation system.
- Up to 32MB data buffer with data transfer rates over 150MBs!
- Auto Switching Power Supply and all cables included.

FW 400+ USB 2.0

FW 800/400+USB 2.0

eSATA+FW 800/400+ USB 2.0

Up to 1.0TB
From $107.99
Plug and Play

OWC Mercury Elite Pro™ RAID

Provide data transfer rates of over 80MB per second! With highest capacity external drive available and Plug & Play simplicity, the possibilities are endless!

- Rugged, machined aluminum enclosure & super quiet operation.
- Ultra-protective shock isolation system.
- Paired 7200rpm drives in every OWC RAID -configured solution.
- Up to 64MB data buffer with data transfer rates over 80MB!
- Custom high performance Oxford Bridge.

Plug and Play

Up to 2.0TB
from $219.99

NewerTech® Guardian MAXimus®

High-Performance RAID 1 'MIRROR' Solution provides "Live Activity" backup/data redundancy of your critical data with Plug and Play simplicity.

- Up to 1.0TB Dual-Drive Solution
- FireWire® 800/400+USB 2.0
- Up to 64MB of drive cache (32MB per drive)
- Ultra-protective shock isolation system.
- Custom Oxford 924 Based Bridge Solution.

Plug and Play

Up to 1.0TB
from $295.99

OWC Mercury On-The- Go™

Safely and Conveniently Transport Large Amounts of Data with No AC Adapter Needed!

- Compact 3.5"(W) x 5.5"(D) x 1"(H) size.
- Bus Powered or may be powered with AC adapter.*
- Ultra-protective shock isolation system.
- Includes carrying case and all cables.
- Custom high performance Oxford Bridge.
- Up to 7200rpm and 16MB data buffer!

Up to 320GB
from $79.99
Plug and Play

FW400 | eSATA+USB 2.0 | USB 2.0 | 'Triple' | 'Combo'

*Use of AC Adapter is OPTIONAL, units are designed for Bus Power and do not require the use of external power source.

OWC Mercury Rack Pro™

Ideal for applications requiring high data throughput, capacity and flexibility.

- Hardware RAID, SoftRaid, JBOD options.
- Available with eSATA, FireWire 800/400, and USB 2.0 interface support.
- User customizable configurations.
- Up to 4.0TB of performance storage.

Up to 4.0TB
from $679.99
Plug and Play

RAID-0 Stripe
RAID-0+1 Stripe & Mirror
FW JBOD

Front

NewerTech® miniStack® v3

Fast, Flexible, and packed with Four Interfaces, the miniStack *v3* offers a techno trinity of performance, convenience, and connectivity making it the ultimate single drive storage solution.

- Compact 6.5"(W) x 6.5"(D) x 1.5"(H) size & super quiet operation.
- 'Quad Interface' FireWire® 800/400/USB 2.0/eSATA
- Up to 32MB data buffer with data transfer rates over 150MBs!
- Oxford OXUF924DSb -with Native Command Queuing (NCQ) for optimum drive performance.

Plug and Play

Up to1.0TB
from $149.99

All pre-configured solutions include backup and disk utilities: • ProSoft Engineering® Data Backup for Mac® OS X v10.2.8 or later • NovaStor NovaBACKUP® for Windows® 2000 or later • Intech® Hard Disk SpeedTools Utilities™ OEM for Mac OS X only

Other World Computing

Visit: www.owcmac.com or Call: 800.275.4576

Award-winning products: Macworld | MacAddict | LAYERS | Photoshop User

Our solutions work with OS X Leopard's Time Machine feature to keep an up-to-date copy of all of your files and documents. Go back in time to recover anything with ease.

BizRate 2007

```
        on setUp()
            copy lib's GuessingGame to TestGuessingGame
            set TestGuessingGame's testNumber to 40
            tell TestGuessingGame to SetNewGuess()
        end setUp
        set testGuess to TestGuessingGame's GetNewGuess()
        set GuessIsHighOrLow to TestGuessingGame's
CheckTheGuess()
        should(GuessIsHighOrLow is equal to ¬
            "You picked the right number!", "wrong guess!")
    end script
```

In this test, we are checking the success condition for when the guess is correct. When we run the tests as before and enter 40 for all of the tests, we get the following results:

```
My Guessing Game Tests

MoreGuessingTests - GuessIsHigherThanRandomNumber ... ok
MoreGuessingTests - GuessIsLowerThanRandomNumber ... ok
MoreGuessingTests - GuessIsCorrect ... ok

Ran 3 tests in 4 seconds.  passed: 3  skips: 0  errors: 0
failures: 0

OK
```

If we use 45 for all the tests, we get the following results:

```
My Guessing Game Tests

MoreGuessingTests - GuessIsHigherThanRandomNumber ... FAIL
MoreGuessingTests - GuessIsLowerThanRandomNumber ... ok
MoreGuessingTests - GuessIsCorrect ... FAIL

FAILURES
----------------------------------------
test: MoreGuessingTests - GuessIsHigherThanRandomNumber
message: guess is too high!
----------------------------------------
test: MoreGuessingTests - GuessIsCorrect
message: wrong guess!
----------------------------------------

Ran 3 tests in 6 seconds.  passed: 1  skips: 0  errors: 0
failures: 2

FAILED
```

The results show that the guess is too high (45 > 40) and that it is the wrong number, so it looks like the tests are working.

We will now develop the feature to keep track of how many guesses are made by the player. As before, we start the development cycle by adding a new test to Listing 6 above:

```
    script |GuessCountIsCorrect|
        property parent : registerTestCase(me)
        on setUp()
            copy lib's GuessingGame to TestGuessingGame
            set TestGuessingGame's testNumber to 40
            tell TestGuessingGame to SetNewGuess()
        end setUp
        -- Do three guesses and check to see if guessCounter =
3
        set testGuess to TestGuessingGame's GetNewGuess()
        set GuessIsHighOrLow to TestGuessingGame's
CheckTheGuess()
        tell TestGuessingGame to IncrementGuessCounter()
        set testGuess to TestGuessingGame's GetNewGuess()
        set GuessIsHighOrLow to TestGuessingGame's
CheckTheGuess()
        tell TestGuessingGame to IncrementGuessCounter()
        set testGuess to TestGuessingGame's GetNewGuess()
        set GuessIsHighOrLow to TestGuessingGame's
CheckTheGuess()
        tell TestGuessingGame to IncrementGuessCounter()
        set checkGuessCounter to TestGuessingGame's
GetCounterValue()
        should(checkGuessCounter is equal to 3, ¬
            "guessCounter has the wrong count!")
    end script
```

When we run our tests using 40 as the input value, we get the following results:

```
My Guessing Game Tests

MoreGuessingTests - GuessIsHigherThanRandomNumber ... ok
MoreGuessingTests - GuessIsLowerThanRandomNumber ... ok
MoreGuessingTests - GuessIsCorrect ... ok
MoreGuessingTests - GuessCountIsCorrect ... ERROR

ERRORS
----------------------------------------
test: MoreGuessingTests - GuessCountIsCorrect
message: ÇscriptÈ doesn't understand the
IncrementGuessCounter message. (-1708)
----------------------------------------

Ran 4 tests in 10 seconds.  passed: 3  skips: 0  errors: 1
failures: 0

FAILED
```

Now we add some functions to the GuessingGame script to increment the counter and get the value of the counter. Also, we will add a new property variable called numberOfGuesses to collect how many guesses the player makes.

```
property numberOfGuesses : 0


on IncrementGuessCounter()
    set numberOfGuesses to numberOfGuesses + 1
end IncrementGuessCounter
on GetCounterValue()
    return numberOfGuesses
end GetCounterValue
```

Repeating our previous tests (entering 40 for each test), we get the following results:

```
My Guessing Game Tests

MoreGuessingTests - GuessIsHigherThanRandomNumber ... ok
MoreGuessingTests - GuessIsLowerThanRandomNumber ... ok
MoreGuessingTests - GuessIsCorrect ... ok
MoreGuessingTests - GuessCountIsCorrect ... ok

Ran 4 tests in 6 seconds.  passed: 4  skips: 0  errors: 0
failures: 0

OK
```

However, in running the tests, the script only asked for a single value for the `GuessCountIsCorrect` test. In looking at the test, we see that we call `SetNewGuess` only in the `setUp` function. To see that we are asking for a new value each time in the test, we need to add a call to `SetNewGuess` before the call

to CheckNewGuess. After adding those calls, we can delete the call to SetNewGuess in the setUp function. Our test now looks like this:

```
    script |GuessCountIsCorrect|
        property parent : registerTestCase(me)
        on setUp()
            copy lib's GuessingGame to TestGuessingGame
            set TestGuessingGame's testNumber to 40
        end setUp
        - Do three guesses and check to see if guessCounter =
3
        tell TestGuessingGame to SetNewGuess()
        set testGuess to TestGuessingGame's GetNewGuess()
        set GuessIsHighOrLow to TestGuessingGame's
CheckTheGuess()
        tell TestGuessingGame to IncrementGuessCounter()

        tell TestGuessingGame to SetNewGuess()
        set testGuess to TestGuessingGame's GetNewGuess()
        set GuessIsHighOrLow to TestGuessingGame's
CheckTheGuess()
        tell TestGuessingGame to IncrementGuessCounter()

        tell TestGuessingGame to SetNewGuess()
        set testGuess to TestGuessingGame's GetNewGuess()
        set GuessIsHighOrLow to TestGuessingGame's
CheckTheGuess()
        tell TestGuessingGame to IncrementGuessCounter()

        set checkGuessCounter to TestGuessingGame's
GetCounterValue()
        should(checkGuessCounter is equal to 3, ¬
            "guessCounter has the wrong count!")
    end script
```

After making this change, the tests return the same results, but it asks for a guess three times in the GuessCountIsCorrect test.

With the addition of the counter functions, we are ready to create our game. To do this, we will create a short script to call the functions we have developed in this article and add it to the GuessingGame script file. Our final script file is shown in Listing 8:

```
script GuessingGame
    property newGuess : 0.0
    property testNumber : 0
    property checkNewGuess : 0.0
    property numberOfGuesses : 0
    property checkTestGuess : 0
    property guessResponse : "test"
    property rangeResponse : "test"
    property checkResponse : "test"
    on SetNewGuess()
        set newGuess to text returned of ¬
        (display dialog "Pick a number from 1 to 100" ¬
        default answer "")
    end SetNewGuess
    on GetNewGuess()
        set guessResponse to CheckGuessCases()
        return guessResponse
    end GetNewGuess
    on CheckGuessCases()
        try
            set checkNewGuess to newGuess as number
            if checkNewGuess is greater than 100 then
                set rangeResponse to ¬
                "Your guess is too high! Pick a number from 1
to 100."
            else if checkNewGuess is less than 1 then
                set rangeResponse to ¬
                "Your guess is too low! Pick a number from 1
to 100."
```

```
            else
                set rangeResponse to "Guess is within range."
            end if
        on error
            set rangeResponse to ¬
            "Your guess is not an integer! Pick a number from
1 to 100."
        end try
        return rangeResponse
    end CheckGuessCases
    on CheckTheGuess()
        if checkNewGuess is less than testNumber then
            set checkResponse to ¬
            "Your guess is too low! Pick a higher number."
        else if checkNewGuess is greater than testNumber then
            set checkResponse to ¬
            "Your guess is too high! Pick a lower number."
        else
            set checkResponse to "You picked the right
number!"
        end if
        return checkResponse
    end CheckTheGuess
    on IncrementGuessCounter()
        set numberOfGuesses to numberOfGuesses + 1
    end IncrementGuessCounter
    on GetCounterValue()
        return numberOfGuesses
    end GetCounterValue
end script

script theGame
    copy GuessingGame to TestGuessingGame
    set TestGuessingGame's testNumber to random number from 1
to 100
    repeat until TestGuessingGame's checkResponse = ¬
            "You picked the right number!"
        tell TestGuessingGame to SetNewGuess()
        set checkGuess to TestGuessingGame's GetNewGuess()
        if checkGuess is not equal to "Guess is within
range." then
            display dialog checkGuess
        else
            set checkGuess to TestGuessingGame's
CheckTheGuess()
            display dialog checkGuess
        end if
        tell TestGuessingGame to IncrementGuessCounter()
    end repeat
    display dialog "You guessed the right answer in " ¬
            & TestGuessingGame's numberOfGuesses & "
guesses!"
end script

run theGame
```

Conclusion

This series has demonstrated some of the principles of test-driven development, and how to use these principles in developing AppleScript programs. By using these techniques, you can develop well-tested code and be able to refactor and update your applications with confidence, knowing that you have a set of tests to ensure that your changes work. I recommend the following resources for using ASUnit in your programs:

The author of ASUnit has a set of tests for his SafeTerminal application at

http://nirs.freeshell.org/code/SafeTerminal/tests/

Matt Neuberg's book, AppleScript: The Definitive Guide is a treasure of information on AppleScript development (http://www.oreilly.com/catalog/applescpttdg2/index.html)

Charles Ross has written an excellent article on using object-oriented programming concepts in AppleScript at http://www.atpm.com/9.02/roll.shtml

Wikipedia's entry on test-driven development covers a wide range of resources at

http://en.wikipedia.org/wiki/Test_driven_development

About The Author

Andy Sylvester is an aerospace engineer who has worked in software development for over twenty years. His interests include web applications, the use of computers in music composition, and software development techniques. He has a weblog at www.andysylvester.com ***where he writes on these subjects. You can reach him at*** andy@andysylvester.com.

The Missing Sync™

Reliable Handheld Synchronization for Mac

Thanks to **The Missing Sync**, Mac users are no longer second-class citizens when it comes to keeping their mobile devices in sync. Whether it's a **Windows Mobile 6** device, a **BlackBerry** or even a **Palm** Treo, there's a Missing Sync product made to connect and synchronize that device with Mac OS X.

Syncs Contacts, Calendars, Tasks

- Supports Address Book, iCal, Microsoft Entourage 2004
- Outstanding field support, even syncs Address Book contact photos
- Supports calendar event reminders and detached events
- Sync Services-savvy for syncing with third-party apps

Complete Notes Syncronization

- Includes Mark/Space Notebook for Mac to create, edit, categorize, sort and search through notes
- Also supports Microsoft Entourage 2004 and Bare Bones Yojimbo
- Full support for Sync Services and .Mac syncing between Macs

iPhoto and iTunes Integration

- Imports photos and videos from mobile devices into iPhoto albums or folders in the Finder
- Resizes and downloads selected iPhoto albums to the device for handheld viewing and slideshows
- Downloads DRM-free music and podcasts for mobile playback

The Missing Sync for BlackBerry, Palm OS or Windows Mobile is available in single-user licenses for $39.95 or in multi-user packs for any size organization, The Missing Sync family of products provide a Mac-centric synchronization solution second to none.

Visit www.markspace.com/reliable today to see how easy it is to sync the latest smartphones and mobile devices with the Mac.

mark/space

The iPhone SDK

Overview and Initial reactions

by Rich Warren

The Announcement

On March 6, 2008, Apple outlined the future of the iPhone. Phil Schiller, Senior Vice President of World Wide Product Marketing, presented new and upcoming enterprise features. Scott Forstall gave us a peek into the much-anticipated iPhone Software Development Kit, and Apple CEO Steve Jobs described the iTunes Application Store.

All of the features will be included in the iPhone 2.0 update, expected in late June. While a free update for the iPhone, iPod Touch owners will have to pay for it. The cost has not yet been announced.

Meanwhile, beta programs are available for both the iPhone SDK and the enterprise features. Apple hopes to get as much feedback as possible before the 2.0 update goes live, so some of the details discussed below might change before the actual release.

Note: While I specifically talk about the iPhone in this article, almost everything applies to the iPod Touch as well.

Enterprise

Even though Apple originally targeted the iPhone at the consumer market, many businesses have expressed an interest in using it in the workplace. Unfortunately, the iPhone lacks several key enterprise features. Basically, corporations need to push data out as soon as it becomes available, whether the data is an important email message, updates to a contact's phone number, or a change in your meeting schedule.

To be honest, as a software engineer, push email confuses me. I mean, email is supposed to be asynchronous, right? They write it at their convenience; I read it at mine. If someone needs an immediate response, they should call me. After all, the word "phone" is right there in the iPhone's name. But...I'm not a sales rep. What do I know?

And I must admit, the iPhone's enterprise features look impressive. Apple has licensed Microsoft's ActiveSync protocol, allowing the iPhone to communicate directly with an Exchange server. This gives enterprise users push email, push calendar, push contacts, access to a global address list and the ability to remote wipe the phone.

Apple will also provide an enterprise developer program. This will allow corporations to develop and distribute proprietary software for internal use.

The SDK

While many people are excited by the Enterprise announcements, clearly the SDK was the star of the show. Last October, Apple announced that they would release the iPhone SDK in February, and developers have held an impatient vigil outside the Cupertino campus ever since.

Let's take a quick look at the iPhone OS and the tools available to iPhone developers. Then we'll try a brief snippet of code, and discuss the path to becoming a real iPhone developer.

The Layers of iPhone OS

Apple introduced the iPhone OS as a series of layers: Core OS at the bottom, and Cocoa Touch at the top.

The Core OS forms the foundation for all other layers, and the OS X Kernel is the foundation for the Core OS. This is basically the same kernel used by Mac OS X. Apple has optimized its performance in low memory situations; however, they still build the iPhone Kernel out of the same project, using the same source code, as the Mac OS X version.

Other Core OS features include: a BSD networking layer, access to Bonjour, and advanced power management.

Power management on the iPhone goes beyond the already impressive systems built into Apple's portable computers. Said Scott Forstall, "The Core OS power manages all of the chips, all of the sensors, the entire operating system, and your application also, automatically."

Core OS

OS X Kernel
Lib System
BSD TCP/IP
Sockets
Security
Power Management
Keychain
Certificates
File System
Bonjour

> "In many ways the iPhone is, first and foremost, an iPod."

Next comes The Core Services layer. This includes a complete set of APIs for accessing many of the workhorse services on the iPhone. While they may not seem flashy and exciting, they make our applications truly useful. Of course, the Core Location API brings a little sparkle and pizzazz of its own.

As you probably already know, the iPhone's Maps application can use cell towers and wireless hotspots to triangulate your position. The accuracy will vary, depending on the number and quality of the signals you can see. Not enough signals, and your location is simply not available. Still, this is an impressive piece of technology.

Now these location-based features are available to third-party developers. The Core Location API will let us build location-aware applications; programs that know our current location, and respond appropriately.

Core Services

Collections
Address Book
Networking
File Access
Core Location
Net Services
Threading
Preferences
URL Utilities

In many ways the iPhone is, first and foremost, an iPod. Not surprisingly, it comes with a rich set of media features.

The Media layer starts with the Core Audio API. This is the same, low-level API used on the Mac OS X. It forms the basis of Apple's consumer and pro audio applications.

Apple then built OpenAL on top of Core Audio. While primarily of interest to game developers, OpenAL provides an industry standard API for rendering multi-channel, three dimensional, positional audio.

Meanwhile, on the visual side, Core Animation provides a powerful set of APIs for easily creating layered animation. While it was initially introduced for Leopard, in many ways Apple built Core Animation for the iPhone. Its user interface heavily uses Core Animation. Almost all the transitions are based on Core Animation effects.

And, OpenGL ES is the embedded version of the popular OpenGL 3D graphics API. This provides a subset of the full OpenGL's functionality, providing a simplified interface with many of OpenGL's redundant or inefficient features removed. This gives us a lightweight API whose performance is comparable to the heavier desktop version.

Media

Core Audio
OpenAL
Audio Mixing
Audio Recording
Video Playback
JPG, PNG, TIFF
PDF
Quartz(2D)
Core Animation
OpenGL ES

Finally, Cocoa Touch replaces the Cocoa UI from Mac OS X.

The mouse and keyboard are gone. Instead, Apple built the iPhone UI around touch as input. The advanced multi-touch event system sits front and center. This handles everything from single finger touches, to multi-finger touches and complex gestures.

This layer also provides access to the camera, and to the iPhone's full, 3-axis accelerometer.

Cocoa Touch

Multi-Touch Events
Multi-Touch Controls
Accelerometer
View Hierarchy
Localization
Alerts
Web View
People Picker
Image Picker
Camera

Development Tools

Now that we understand the basic layout of the iPhone OS, how do we manipulate it? Well, iPhone developers use four main tools: Xcode, Interface Builder, Instruments and the iPhone Simulator. Every Mac developer should instantly recognize the first three. They are the standard development tools used for Mac OS X. Now, Apple has enhanced them to support iPhone projects. The Simulator, on the other hand, is an iPhone specific addition to the developer's toolbox.

Xcode is a powerful IDE for building and managing both Mac and iPhone projects. The new version of Xcode includes three templates for iPhone applications: Cocoa Touch Application, Cocoa Touch List and Cocoa Touch Toolbar.

As you would expect, all of Xcode' standard features still work when developing for the iPhone. For example, code completion recognizes for both the Mac and iPhone APIs.

Additionally, if you've ever done embedded development, then you understand that the development cycle can be painful. First you compile the application. Then you transfer the application onto the device. Finally, you launch and test the application. Often, each step requires a separate tool.

Xcode reduces this to a single click. Just press the Build and Go button, and Xcode will compile your application, transfer it to either the Simulator or to your iPhone (depending on the target settings) and launch the application. More importantly, the Xcode debugger allows you to remotely debug your application while it runs on either the Simulator or on the iPhone itself.

Next we have Interface Builder—and a complete library of iPhone-specific controls. Interface Builder allows you to visually lay out and design your user interface. You can draw the connections between your objects and your interface. It is even a powerful localization tool.

Instruments provides a powerful suite of performance analysis and visualization tools, based on the DTrace analysis engine. Just like the debugger, Instruments allows you to remotely monitor applications running on either the Simulator or the iPhone. You can record a wide range of system behaviors, from network and file activity to CPU sampling and memory usage. Different elements are shown on parallel tracks, allowing you to quickly and easily spot correlated issues. Instruments can also access the stack trace, giving you access to the code executing at a given point in time.

Finally, the Simulator (as the name suggests) simulates an iPhone. It supports the entire API stack for the iPhone OS. It also comes with a few built-in applications, including a complete build of Safari. Yes, you can use the simulator to test web applications as well.

Hello World

All right, we've seen the OS and we've seen the tools Let's look at a simple application. First, you will need to download and install the iPhone SDK.

Once that's done, open Xcode. Select File -> New Project.... Then choose Cocoa Touch Application. This will create a blank iPhone application.

Now, open the `MyView.m` file. Insert the following code:

MyView.m

A mandatory Hello World application for the iPhone.

```
@implementation MyView

- (void)drawRect:(CGRect)rect {

    [[UIColor whiteColor] set];
    [@"Hello World"    drawInRect:CGRectMake(0, 175, 320,
50)
                        withFont:[UIFont
 fontWithName:@"Marker Felt" size:50]

lineBreakMode:UILineBreakModeMiddleTruncation
                        alignment:UITextAlignmentCenter];
}

@end
```

This draws the words "Hello World" in white, and centers it in the iPhone's screen.

Now, click Build and Go. This will compile your application and launch it in the iPhone simulator. That's it, an obligatory "Hello World" application for the iPhone.

Becoming an iPhone Developer

Everything seems so exciting, right? "But", I can hear you asking, "How do I become an iPhone developer?"

Apple has created a two-step process. First, you must register as an iPhone developer. This is free. Just go to http://developer.apple.com/iphone/program, and click the Apply button.

Registering not only allows you to download the beta SDK, it also gives you access to all the developer resources Apple has placed online. Take a look at http://developer.apple.com/iphone. This is your one-stop information depot for all things iPhone or iPod Touch.

Apple has released an impressive array of information. There are ten Getting Started videos, from "Introduction to the iPhone SDK" to "Leveraging iPhone Location, Acceleration, Orientation and System Information." Apple has also posed five Getting Started documents, ranging from an overview of the iPhone OS, to an Objective-C primer, and an iPhone SDK FAQ.

Next, we have a selection of 15 Coding How-To's that dig into more specific topics. These include: handling multi-touch events, recording audio, triggering vibration, and writing secure code.

Apple even presents 13 sample projects. These demonstrate a variety of techniques and their actual use in applications. Again, we have a wide range, from the HelloWorldClassic to AccelerometerGraph, LaunchMe, FingerSketch and GLGravity.

Finally, Apple has included dozens of links into the iPhone Reference Library.

So, registering as an iPhone developer allows you to download the SDK, build and run applications on the simulator and access all the resources listed above. But, if you want to go further, you need to join the iPhone Developer Program.

The iPhone Developer Program costs $99.00. This allows you to build and run applications on the iPhone itself. You can also submit applications to the iTunes Application Store. Unfortunately, Apple is only accepting a limited number of developers during the Beta period. As I write this, the initial slots have already been filled, and Apple has a considerable waiting list.

Don't Forget Web Applications

While the new SDK was the obvious focus of the March 6th event, Apple also made quite a fuss over the existing iPhone web applications. Over 1,000 applications so far, and counting. Despite the new SDK, the web apps are not going anywhere.

"We've really been investing in web applications, and they've been incredibly successful. In this next release, we're going to add even more features to make the experience even better." Said Scott Forstall.

Expect good things for web apps in iPhone 2.0.

One more thing...

In true Steve Jobs style, the March 6th event had one last surprise. The venture capital firm Kleiner Perkins Caufield & Byers announced the creation of the iFund, a $100M investment initiative to fund projects for the iPhone and iPod touch. "That should be enough to start about a dozen Amazons or even four Googles," said John Doerr, Partner at KPCB, "and if we're running out of money, we'll look around to try and find some more."

KPCB sees the iPhone and the iPod Touch as a new, transformative platform. "Think about it. What the iPhone's all about is, in your pocket, you have something that's broadband and connected all the time. It's personal. It knows who you are and where you are. That's a big deal, a really big deal," said John Doerr, "It's bigger than the personal computer."

KPCB plans to fund everything from small startups to large expansions. Their focus areas include location-based services, social networks, mCommerce, communication and entertainment. They are currently accepting applications online.

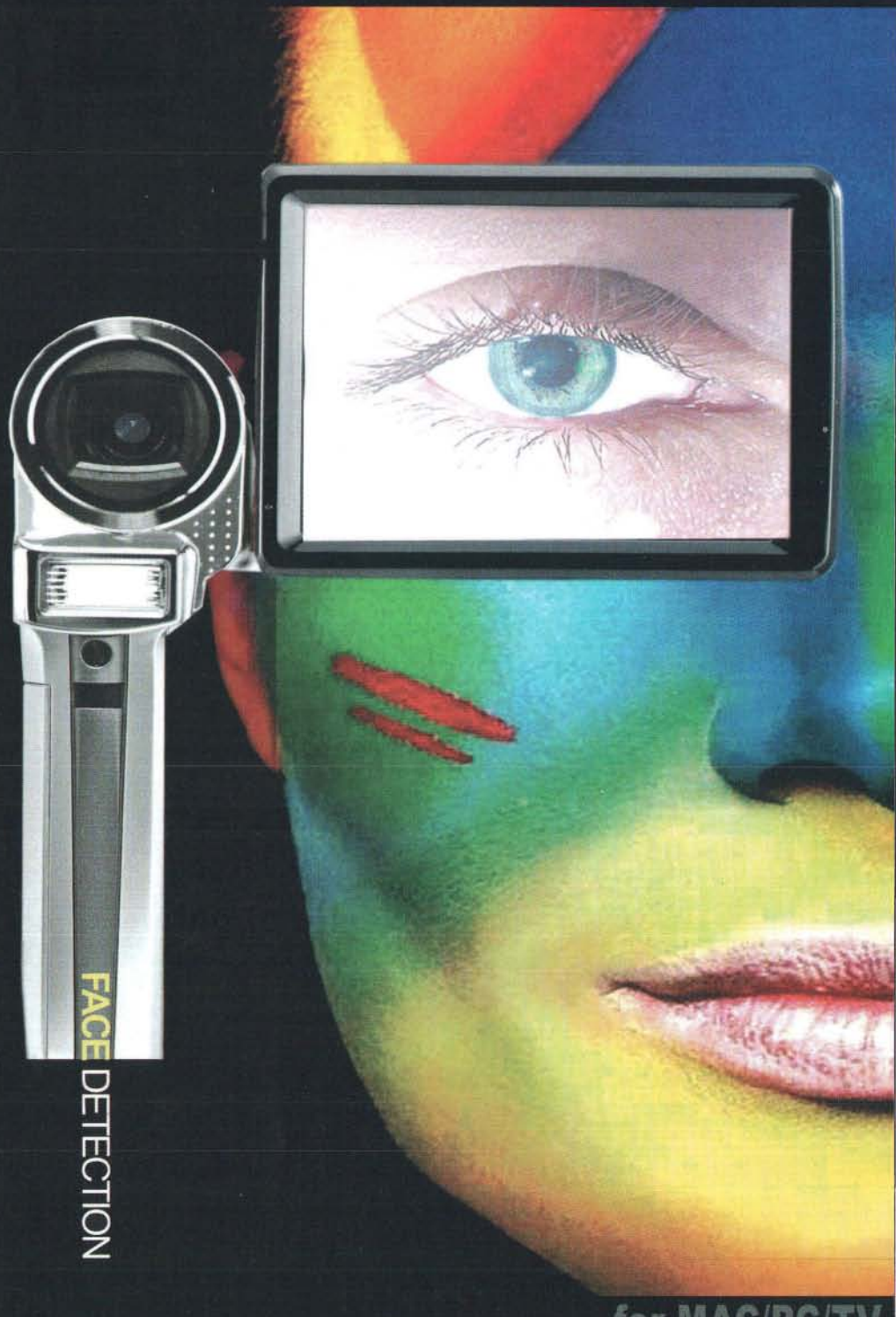
Check out http://www.kpcb.com/initiatives/ifund/index.html for more information.

Public Response

Immediately following the beta release, the Internet filled with the groans, cries and gnashing of teeth. Many technopundits began complaining almost immediately, their glasses apparently stuck at half-empty. Most of the complaints focused on restrictions in the SDK. I will explore some of their arguments and counter-arguments below.

supacam.com

The ONLY camera that streams DVD Quality Video across the internet



for MAC/PC/TV

supastream.com

"The full VGA streaming gives a picture **that brings Video sharing to a new height"**

Professor Norman Medoff author Portable Video ENG

Apple's going to take 30%

Actually, relatively few people have complained about Apple's cut. Similar programs often take a much larger slice of the pie. Handango, for example, takes 40% off the top. But, we still have a lot of unanswered questions about the App Store. We know we can offer free software, and we can provide updates. But, how can we handle downloadable demos? Can we offer special promotions or bundles? Can we give away free review copies for promotional purposes?

This starts to pick at the real issue. While developers largely won't argue over the price, they do complain about the exclusivity. We have no choice, if we want to distribute our applications, we must go through iTunes.

Apple fully intends to vet all our applications, but we don't yet know how restrictive they will be. After all, restrictions for stability, security and safety are probably for the best. No one wants an unstable phone. But, will Apple restrict applications for purely business reasons?

Amazon's online music store directly competes with iTunes. What if Amazon built an iPhone application for purchasing their music directly from the iPhone. Would Apple allow this?

On the other hand, assuming our applications are legal, porn-free and play well with others, Apple should give us their blessing. This puts us in the App Store—in the one and only place people can go to look for iPhone applications. Everyone will come to this watering hole.

The App Store should become an excellent resource, especially for smaller developers.

The App Store should become an excellent resource, especially for smaller developers. We don't need to set up an online store, or manage credit cards. We just put the application up, and collect our checks at the end of the month.

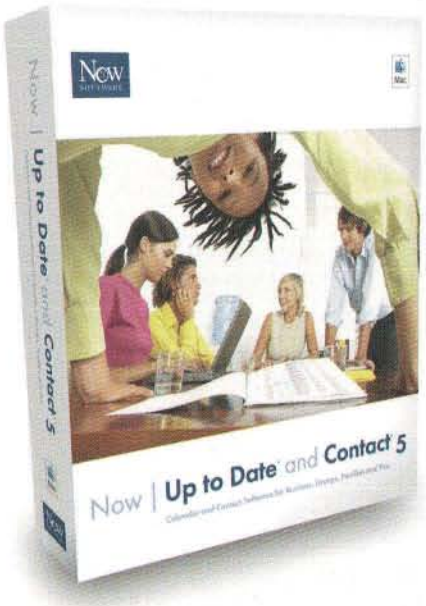
Of course, the store may go the way of the iTunes Podcast directory. Bigger developers might quickly grow to dominate, pushing aside newcomers and smaller teams.

It will be interesting to see how this actually plays out, but I suspect smaller developers may need to market their applications outside of iTunes, or they will just get lost in the crush.

Now **scheduling** and **contact management** for your entire organization.

Now | **Up to Date**® and **Contact**® **5**

Calendar and Contact Software for Business, Groups, Families and You.



Is this project on schedule? When are you available to meet about the systems upgrade? Where are all the field techs today? When was the last time anyone talked to our biggest customer?

Virtually all groups live (or die) by their abilities to meet deadlines and keep track of their customers, prospects, and vendors. Few small companies or even departments of big companies have the tools they need.

Now Up-to-Date & Contact might just be the calendar and contact software for you. It's time-tested and used by more Mac-based companies than any other solution. And it's cross-platform–available for your PC users, too. It's easy to install and manage and simple for your employees to understand and use.

Using Now Up-to-Date & Contact you can schedule meetings for multiple users, view multiple, simultaneous calendars, and reserve rooms and resources. You can share contact information about your customers, prospects and vendors. And using our free server software you can set it up in minutes and share with users in the office or from anywhere with an internet connection.

NOW SOFTWARE

Phone: 866-527-0556

Web: www.nowsoftware.com

Call us now at 866-527-0556 or email us at mactech@nowsoftware.com and we'll send you our free evaluation kit, including the book that will make it all easy, "Take Control of Now Up-to-Date & Contact" from Take Control books!

Only one application at a time

The lack of background processes has become a major source of complaints.

Put simply, we cannot run our applications in the background. Our application will shut down as soon as we open something else. This is particularly troublesome, since outside events can trigger shutdowns. As soon as a call comes in our application quits.

Of course, most programs should just save their state before closing, then reload when the user next needs them. For many applications, this will seem nearly transparent.

A few applications, however, really benefit from running in the background. The AIM client is an often-cited example. Every time your phone rings, or every time we switch to a different application, the AIM client shuts down. When it shuts down, it logs us off.

On the plus side, the lack of background processes prevents a wide range of potential problems—from spyware to memory leaks, from battery life to stability. Having one application at a time really limits the damage that we can do to ourselves.

No plugins or interpreted code

Apple has clearly stated that they will distribute iPhone applications exclusively through their store. They want a chance to vet all incoming code, and the ability to monitor and track any malicious or problematic applications.

Plugins and interpreters offer a back door to the iPhone. They would let developers sidestep Apple's controls.

On the surface, the restriction against plugins and interpreted code doesn't seem like a big deal, but many popular desktop applications, like FireFox or Adobe Photoshop, rely heavily on plugin architectures. Of course, those developers could simply limit plugins for the iPhone version—provide a bundle that pre-loads the most popular plugins, and disable the ability to add additional ones.

More to the point, mobile applications should feel more focused than their desktop siblings. They should provide a single solution to a single, tightly defined problem. In many ways, plugin architectures run contrary to this design goal.

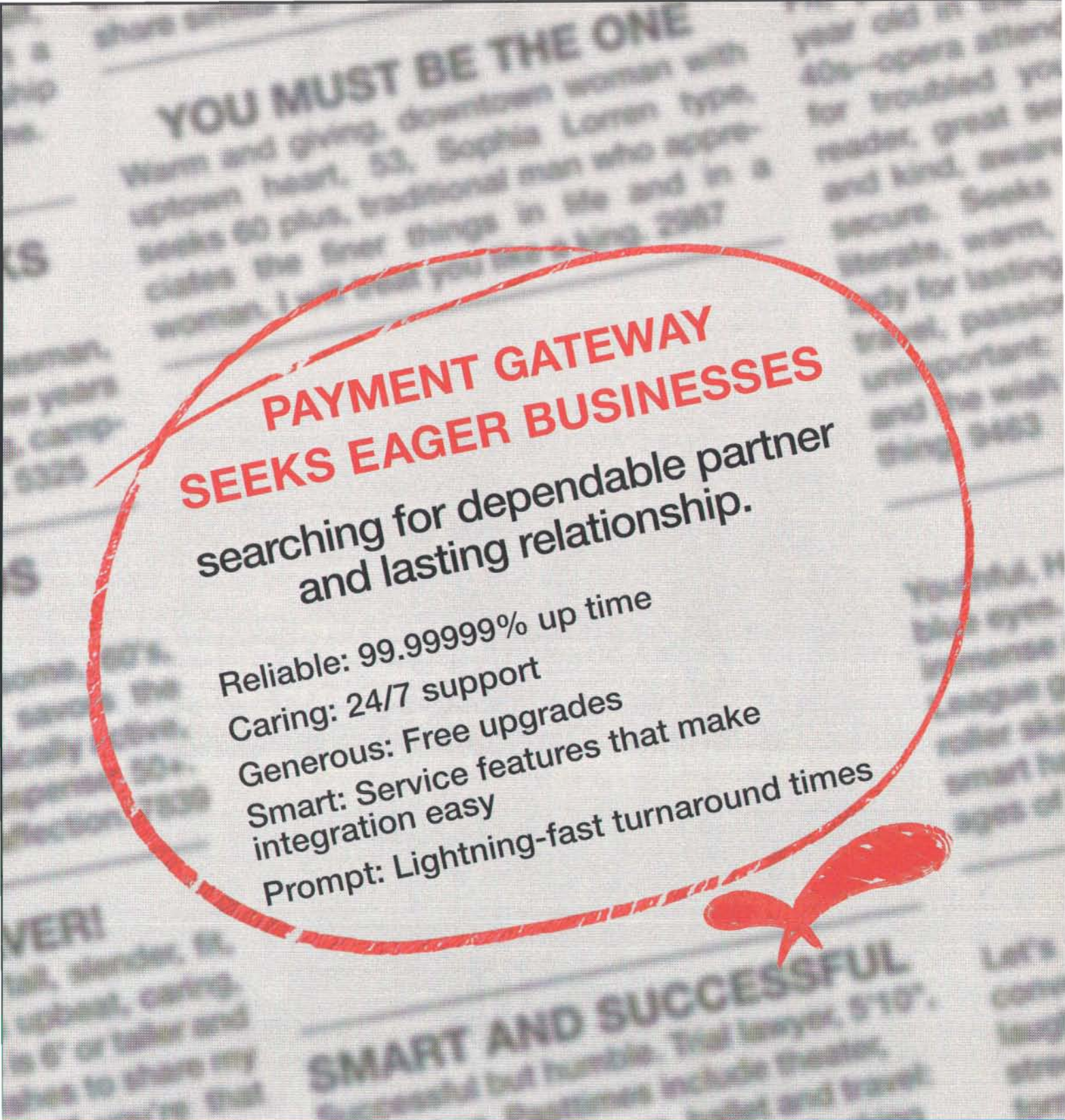
No VoIP over the cellular network

I have trouble believing that this really surprised anyone. Obviously, Apple's cellular partners would prefer that we used (and paid for) our cellular minutes, and Apple needs to protect their partners, at least to some degree. To me, this was a no brainer—kind of like asking whether Apple would restrict jail breaking applications.

Most people seem to agree. Many felt that limiting VoIP to wireless networks was not only reasonable, but completely expected. Most were just happy to have wireless VoIP.

I have to code in Objective-C? On a Mac?

As a Mac user and developer, it's easy to forget how deeply the iPod and iPhone have penetrated into the Windows



Let Us Connect You with the perfect match for your business.

A payment gateway is an inexpensive and automated solution for handling payment transactions "live" with a secured Internet connection. LUCY Gateway™ is a payment processing engine that provides a secure bridge between your Point-of-Sale system and the back-end clearance. LUCY Gateway securely processes all forms of electronic payment—credit cards, PIN debit, checks, gift cards, EBT and signature capture.

www.cynergydata.com/gateway 866.414.3690

Cynergy Data, LLC is a registered ISO/MSP in association with Bank of America, N.A., Charlotte, NC

lucy™ Gateway
let us connect you
A Cynergy Data Solution

world. While I'm excited about the possibility of developing iPhone apps with Objective-C on my Mac, For many, this may be a strange and somewhat frightening experience. But, really. What did you expect.

Where is Interface Builder?

This seems more like disappointment than a complaint. After seeing the Interface Builder in action, many developers eagerly downloaded the SDK, opened their new version of Xcode, and then realized that the iPhone version of Interface Builder was still missing.

Beta Release. Look up the definition if you have any questions.

The Silent Majority

While there is some validity to these complaints, they haven't stopped developers from flocking to the iPhone SDK. The numbers speak for themselves. According to Apple, over 100,000 people downloaded the SDK in the first four days.

An overwhelming number of developers also applied for the iPhone Developer Program. I mean that literally, Apple was overwhelmed. While we don't have hard numbers, it took Apple several days to process the initial applications. After waiting patiently, a number of developers (including yours truly) received brief emails. These explained that the initial number of beta developers had already been filled, and that we would be notified as additional slots became available. Three days to get a canned email response—that's practically a denial of service attack.

Conclusion

The iPhone 2.0 release will truly mark a new era for the iPhone. Apple will release a number of improvements to the iPhone OS itself, including the new Enterprise features. They have also announced parental controls, which can limit access to applications like the App Store or Safari. Additionally, We should see improved support for web applications. Most importantly, however, third-party developers will release a wide range of new applications for our phones.

The iPhone SDK has sent waves of excitement rippling through the developer community. Sure, there are a few unrepentant critics, but most of the responses have been overwhelmingly positive.

The iPhone provides a unique, advanced platform with many intriguing features. Once our applications get Apple's approval, the Application Store gives us instant access to a worldwide market of iPhone and iPod Touch owners.

Finally, the iFund promises to create a collection of new companies and communities centered around the iPhone.

I can hardly wait to see what June brings.

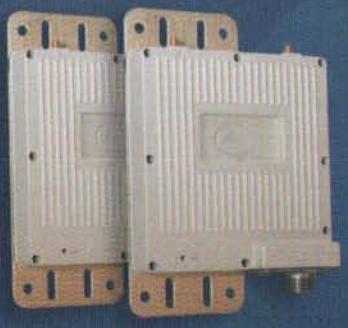
About The Author

Rich Warren lives in Honolulu, Hawaii with his wife, Mika, daughter, Haruko, and his son, Kai. He is a software engineer, freelance writer and part time graduate student. When not playing on the beach, he is probably writing, coding or doing research on his MacBook Pro. You can reach Rich at rikiwarren@mac.com, or check out his blog at http://freelancemadscience.blogspot.com/

WE OUTPERFORM OUR COMPETITION

Nobody beats us on...

- **Price**
  Save thousands on multi-location T1 services
- **Performance**
  6 to 30× faster than T1 service
- **Reliability**
  Best-in-class fixed wireless solutions



Just one of many wireless solutions from Trango...

ATLAS 5010 Series™
Wireless Ethernet Bridge

- 45 Mbps of sustained throughput
- Capable of 5.3 and 5.8 GHz
- Point-to-point, OFDM
- Up to 40-mile range
- Adaptable Rate Modulation

Wireless Connectivity IS Powered by Trango

Trango Broadband Wireless
A division of Trango Systems, Inc.

www.trangobroadband.com
Phone: +1 (858) 653-3900

trangobroadband WIRELESS

by Dave Dribin

Cocoa Puffs

Views, controls, actions, notifications, and delegates

Last month in *The Road to Code* we wrote our first Cocoa GUI application. This month, we're going to explore some of the Cocoa classes working behind the scenes. Every application has different needs, so we're also going to show how Cocoa provides ways to customize its behavior. To recap from last month, we created a simple rectangle calculation program. The resulting user interface (UI) is shown in Figure 1.

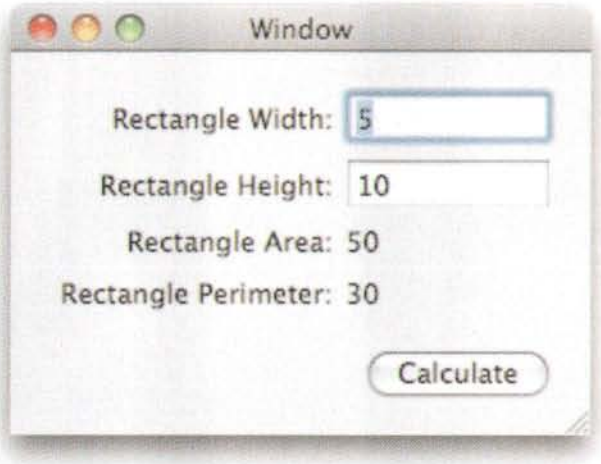


Figure 1: "Hello World" rectangle application

Remember that our application consists of two classes, `Rectangle` and `HelloWorldController`. The `Rectangle` class represents a geometric rectangle and is responsible for the area and perimeter calculations. The interface for the `Rectangle` class is shown in Listing 1. The `HelloWorldController` class has outlets and actions that are attached to various parts of the user interface. It responds to the user clicking on the Calculate button by updating the area and perimeter. Its interface is shown in Listing 2. The user interface itself is stored in a nib file and created by Interface Builder. For the full source code, please download the project file from last month's article.

Listing 1: Rectangle interface

```
#import <Foundation/Foundation.h>

@interface Rectangle : NSObject
{
    float _leftX;
    float _bottomY;
    float _width;
    float _height;
}

@property float leftX;
@property float bottomY;
@property float width;
@property float height;
@property (readonly) float area;
@property (readonly) float perimeter;

- (id) initWithLeftX: (float) leftX
             bottomY: (float) bottomY
              rightX: (float) rightX
                topY: (float) topY;

@end
```

Listing 2: HelloWorldController interface

```
#import <Cocoa/Cocoa.h>

@class Rectangle;

@interface HelloWorldController : NSObject
{
    IBOutlet NSTextField * _widthField;
    IBOutlet NSTextField * _heightField;
    IBOutlet NSTextField * _areaLabel;
    IBOutlet NSTextField * _perimiterLabel;

    Rectangle * _rectangle;
}

- (IBAction) calculate: (id) sender;

- (IBAction) textFieldAction: (id) sender;

@end
```

So what makes this application tick? How does it work? What other classes are involved in running a Cocoa application?

Object-oriented programming maps nicely to GUI programs because a class typically represents each visual GUI component, and Cocoa is no different. By looking at our running application, you can see that it has three different kinds of GUI components: six labels, two text fields, and a button. You know from our outlets that the text fields and labels are both of a class called `NSTextField`. Labels are essentially just read-only text fields. The button is implemented by the `NSButton` class. The `NSTextField` and `NSButton` classes provided by Cocoa are called *controls*. A control is a GUI component that has three primary responsibilities:

Handles user interaction (keyboard and mouse events)
Draws itself on the screen
Sends action messages

Each control handles these responsibilities as appropriate for the data it displays. For example, `NSButton` draws the oval around the text. It also handles mouse clicks by updating its look accordingly and sending an action to the target. Remember that we hooked up the action of the button to the `calculate:` method of our controller. `NSTextField` is a control to display simple, one-line text. For an editable text field, like those representing the height and width, the text field handles keyboard events so that the user can edit its contents. Text fields also have actions, even though we didn't use them. A text field sends its action when the user has finished editing it, either by hitting **Return** or by changing focus from the selected text field. As an exercise, let's add this action to our controller:

```
- (IBAction) textFieldAction: (id) sender
{
    NSLog(@"Text field action: %@",
          [sender stringValue]);
}
```

Now, hook this up to both editable text fields for the width and height in Interface Builder and run the application. If you click between the two text fields, you should see output in the console. In our application, we don't need these text field actions, but the point is that all controls have actions which you can attach to.

To gain further understanding about how buttons and text fields work, we need to examine the inheritance hierarchy for `NSButton` and `NSTextField`, as shown in Figure 2. We'll start at the top and work our way down.

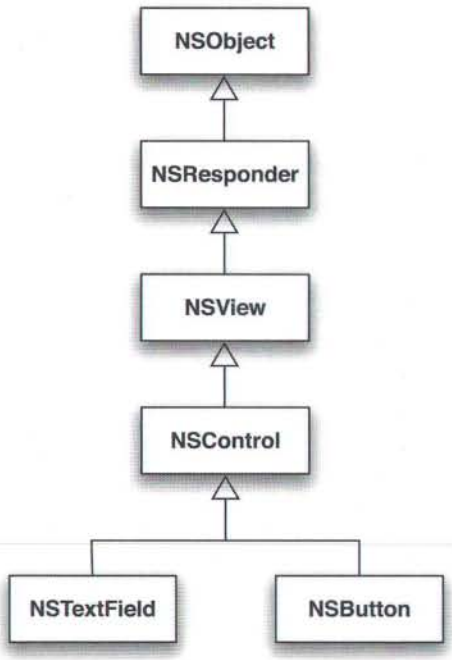


Figure 2: Control hierarchy

First, let's look at `NSObject`. Remember that all classes have `NSObject` at the root of their family tree, and `NSButton` and `NSTextField` are no different. `NSObject` provides basic methods for all Objective-C objects such as `alloc`, `init`, and `release`. All of our objects have used `NSObject` as their parent, too.

Just below `NSObject` is `NSResponder`, and it's responsible for handling user events. Any class that needs to interact with user events, such as keyboard and mouse events, must inherit from `NSResponder`. `NSView` is next, and it's responsible for drawing to the screen. Because it inherits from `NSResponder`, it can also handle user events. Finally, we get to `NSControl`, which adds support for actions. Thus, the three responsibilities of a control are really implemented by three separate classes: `NSResponder`, `NSView`, and `NSControl`.

Apart from `NSTextField` and `NSButton`, there are two other Cocoa classes that make up our application: `NSWindow` and `NSApplication`. Both of these classes have `NSResponder` as their direct superclass, as shown in Figure 3.

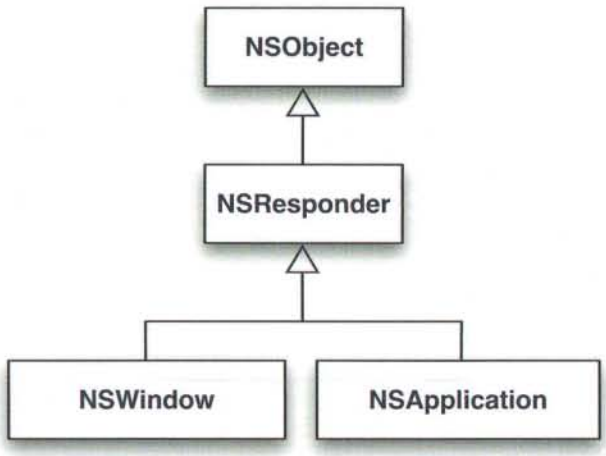


Figure 3: Window and application hierarchy

The `NSWindow` class represents a window. It handles the window events, such as closing and maximizing, and is essentially a container of views. A view must be part of a window to be displayed. In fact `NSView` has a `window` method that returns the `NSWindow` that contains it. An application may have more than one window, but only one window is active it a time. The active window receives user events and is called the *key window*.

The `NSApplication` class represents a Cocoa application. There is always exactly one instance of this class in any running Cocoa application, and it can be accessed one of two ways. The first is the `sharedApplication` class method, and the second is the `NSApp` global variable:

```
// Two ways to get the application instance
NSApplication * application;
application = [NSApplication sharedApplication];
application = NSApp;
```

I usually use the `NSApp` global, as it's shorter to type. The `NSApplication` instance is responsible for running the application's main event loop. It handles all events sent by the underlying operating system and routes them to the correct `NSResponder`. The `NSApplication` class also keeps track of all windows and which window is the key window. You can use `NSApplication` to customize an application's behavior, as we will see later.

How Actions Work

We used an action to customize what happened when the **Calculate** button is pressed. Actions are possible due to unique features of the Objective-C language. GUI toolkits based in other languages might require subclassing or other verbose language syntax to customize the button pressed behavior. Subclassing can still be used to customize the behavior of an existing class, but it should generally be the last resort. Subclassing can make code more difficult to maintain in the future.

Remember that actions have a method signature as follows:

```
- (IBAction) actionName: (id) sender;
```

How *does* Interface Builder hook this up to a control, and how *does* a control call this action? It uses an Objective-C feature called *dynamic dispatch*. Dynamic dispatch is the fancy name for how the Objective-C language calls methods. Dynamic dispatch means Objective-C performs method dispatching at runtime, instead of static dispatch at compile time like C++ or C.

While the compiler handles most of the work of translating Objective-C into machine language, part of the language that is linked into every Objective-C program is called the *Objective-C runtime*. Among other things, the Objective-C runtime does the actual method calling. When you call a method, like this:

```
[anObject aMethod];
```

the object you are calling a method on is called the *target*, and the method name is called the *selector*. Remember that I said calling a method on an object is sometimes called sending a message to an object? Well, think of the target as the recipient of the message and the selector as the message being sent. Under the hood, the compiler turns this method call into a C function call:

```
objc_msgSend(anObject, @selector(aMethod));
```

The `objc_msgSend` function is part of the Objective-C runtime that implements calling methods by sending selectors to target objects. We're using the `@selector` keyword to specify a selector from a method name. It can actually be a bit dangerous to call methods using selectors directly with `objc_msgSend` unless you know what you are doing. However, `NSObject` provides a way to call a method from a selector via the `performSelector:` method. This example shows how to call `aMethod` using this method:

```
[anObject performSelector: @selector(aMethod)];
```

You won't see code like this written most of the time. It is a lot more to type, it is not nearly as readable, and is not as efficient as just calling the method directly. But dynamic dispatching of selectors gives the programmer a lot of flexibility in calling methods of objects. Button actions are implemented using selectors, as we shall see shortly.

Selectors even have their own type, `SEL`, so you could alternatively call this method as follows:

```
SEL selector = @selector(aMethod);
[anObject performSelector: selector];
```

One point about selectors is that you must include the whole method name, including the parameter components. Thus, if you have a method that takes two arguments:

```
- (void)    someMethodWithObject1: (NSObject *) object1
                      andObject2: (NSObject *) object2;
```

the selector would be:

```
SEL selector =
        @selector(someMethodWithObject1:andObject2:);
```

You *must* include the colons in the selectors, too. A common error in dealing with selectors is to forget colons, especially the last one. When you're debugging code that uses selectors, be sure to check for them.

Now, how does the selector business fit in with actions? If you look at the reference documentation for `NSControl`, you will find these four methods:

```
- (void) setTarget: (id) anObject;
- (id) target;

- (void) setAction: (SEL) aSelector;
- (SEL) action;
```

When you control drag from a button to your controller to hook up the action, Interface Builder calls these methods behind the scenes. It calls `setTarget:` with your controller object and `setAction:` with the selector of your action method. Then, when the user clicks on your button, it calls your action method by using a variant of the `performSelector:` method above. Remember that an action method has one argument, which is the calling control. Somewhere inside `NSControl`, there is code that looks similar to this:

```
// Call the action method
[[self target] performSelector: [self action]
                    withObject: self];
```

Notice that it uses the `performSelector:withObject:` method to pass an object to the action method. This is why action methods take a single parameter of type `id`.

The id Data Type

Oh, and there's that pesky `id` type again. It's time to finally come clean about it. The `id` type is a special data type that can point to any Objective-C object. It's similar to `NSObject *`, since most objects derive from `NSObject`, but even more generic. It *is* possible to create a class that does not inherit from `NSObject`, and the `id` type can point to those, too. One benefit is a matter of type safety: you can convert any object from an `id` to another Objective-C class without a cast. For example, this code compiles without any compiler warnings:

```
id object = ...;
Rectangle * rectangle = object;
```

On the other hand, this code will get a compile warning:

```
NSObject * object = ...;
Rectangle * rectangle = object;
```

This just means that you have to use a cast to tell the compiler that you really do want to use `object` as a `Rectangle`:

```
NSObject * object = ...;
Rectangle * rectangle = (Rectangle *) object;
```

So why use the `id` type if all objects are inherited from `NSObject`? It's handy when you don't want to use a cast. One good example is the `objectAtIndex:` method of `NSArray` we used a few months back. It returns an `id`, and thus we don't need to use casts every time we access the elements of an array. If it returned an `NSObject *`, it would mean a lot of useless typing. It's also useful for the `sender` argument of action methods because we don't know the type of object sending the action. It could be any one of the `NSControl` subclasses or even an `NSMenuItem`. Using `id` allows us to skip the casts when converting to the proper type.

Notifications

While Interface Builder usually hides the use of selectors for actions, other parts of Cocoa require using selectors directly. However, actions are just one way to customize Cocoa without subclassing. Another way to customize Cocoa is through *notifications*. Notifications are a way to broadcast an event to any object that is interested. Think of notifications as an object picking up a megaphone and announcing that a certain event occurred to the whole application. Any object that is interested in this event can listen for it and do whatever it wants. Selectors are used for notifications, too. For example, `NSApplication` will send out a notification every time it becomes inactive, and we can register a method to be called (via selectors) whenever this notification is posted.

Each notification must have a unique string name, and objects that are interested in a particular notification must use this name when they register. The virtual megaphone, if you will, is called the notification center and is aptly implemented by a class named `NSNotificationCenter`. To receive a notification, an object must register itself with a notification center.

The notification named `NSApplicationWillResignActiveNotification` is posted by the shared `NSApplication` instance when the application becomes inactive. We can modify our `awakeFromNib` method to register our controller as an observer for this notification as follows:

```
- (void) awakeFromNib
{
    [_widthField setFloatValue: _rectangle.width];
    [_heightField setFloatValue: _rectangle.height];
    [self updateAreaAndPerimeter];

    NSNotificationCenter * center =
        [NSNotificationCenter defaultCenter];
    [center addObserver: self
               selector:
@selector(applicationWillResignActive:)
                   name:
NSApplicationWillResignActiveNotification
                 object: NSApp];
}
```

There are other notification centers, but usually the default notification center is used, as in this case. The documentation should be clear about which notification center to use, if you have any doubt. You can see that this registration method uses selectors to tie a particular notification to a method.

The `name` parameter is the name of the notification we are interested in observing. The `object` parameter is the source of the notification. In our case, there is only one instance of `NSApplication`, but other notifications may be tied to a specific instance of an object. For example, `NSWindows` send out notifications when they close, and you may be interested only in a particular window instance closing, not all windows. The `observer` parameter is the target of a method call, and the `selector` parameter is a specific method on the target to call when the notification occurs. The end result of this is that the `applicationWillResignActive:` method of `HelloWorldController` will get called every time the application becomes inactive. Let's fill in a simple implementation of this method:

```
- (void) applicationWillResignActive: (NSNotification *)
notification
{
    NSLog(@"Application will resign active");
}
```

If you now run the application, you should see this log statement printed every time you bring another application to the front. `NSApplication` has a bunch of other notifications that you can use, as well. See the reference documentation for full information.

Oh, and you must not forget to unregister with the notification center, when you no longer want to receive notifications. Like calling `release` for every `retain`, you need to unregister for every notification you register for. This is especially important for non-garbage collected applications. If you don't unregister for notifications, the notification center will try to call methods on a non-existent object, which can lead

to runtime errors. Thus, it is a good idea to unregister in the `dealloc` method:

```
- (void) dealloc
{
    [_rectangle release];

    NSNotificationCenter * center =
        [NSNotificationCenter defaultCenter];
    [center removeObserver: self];
    [super dealloc];
}
```

This removes an object for all notifications, which is handy if you've registered for multiple notifications on multiple objects. You don't need to do this cleanup for garbage collected applications as the notification center will automatically unregister all notifications for you when your object is garbage collected.

Delegation

Notifications are good for broadcasting events to multiple objects, but sometimes events need more one-on-one attention. Another way to customize behavior of certain Cocoa classes is through *delegation*. Just like managers delegate decisions to their employees, delegation is a technique used where one object delegates certain decisions to another object.

Let's stick with `NSApplication` as an example. You may notice that some applications automatically quit when their last window closes to prevent the application from sticking around unnecessarily. This is actually a simple customization of `NSApplication` through delegation.

The `NSApplication` object contains a reference to another object called a *delegate*. The delegate may customize certain actions of the application. For example, when the last window of an application closes, `NSApp` calls this method on its delegate:

```
- (BOOL)applicationShouldTerminateAfterLastWindowClosed:
    (NSApplication *)theApplication;
```

If the delegate returns `YES`, the application quits. Otherwise it continues running. The interesting thing about delegates is that there may be many delegate methods available for customization, but the delegate only needs to implement the ones it needs to customize. In this case, if the delegate does not implement this method, or there is no delegate, then the default response is `NO`. This is why our application currently will not quit if we close the window.

Like actions, delegates for most GUI classes can be established in Interface Builder. I'm going to go through the procedure to setup our `HelloWorldController` as the delegate to `NSApplication` in order to customize this behavior. Just like our controller class has outlets that can be connected through Interface Builder, `NSApplication`'s delegate is also an outlet that can be hooked up through Interface Builder. Double click on `MainMenu.nib` to fire up Interface Builder. In the MainMenu.nib window, you should see an icon representing the application. Control-drag from this

icon to your controller icon. Interface Builder should prompt you to connect an outlet by popping up a window. Choose delegate from the menu, as shown in Figure 4.

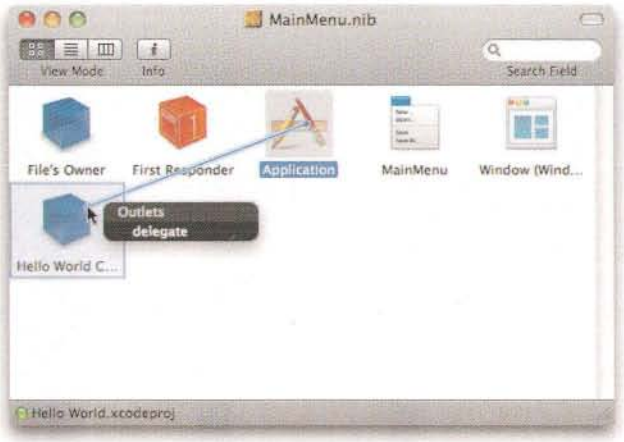


Figure 4: Connecting the delegate outlet

This makes our controller the delegate of the application, but it does not yet override any of the delegate methods. You need to add this method to your controller implementation:

```
- (BOOL)applicationShouldTerminateAfterLastWindowClosed:
    (NSApplication *)theApplication
{
    return YES;
}
```

If you now run our application, it should quit when you close the window.

Delegates and Selectors

You may be asking how, exactly, do selectors fit into delegation? It's a bit more behind-the-scenes than actions and notifications, but it's useful to know if you want to setup your own delegates. As you've seen, you must implement a specific method for delegation to work. You cannot customize which method as you can with notifications, for example. It is sort of like subclassing `NSApplication` and overriding a method without the hassle of creating a new subclass. If you step into the shoes of the person who wrote `NSApplication`, you would see the use of selectors again. Here's how it would be implemented:

```
BOOL shouldTerminate = NO;
id delegate = [self delegate];
SEL selector =

@selector(applicationShouldTerminateAfterLastWindowClosed:);
if ([delegate respondsToSelector: selector])
{
    shouldTerminate = [delegate
        applicationShouldTerminateAfterLastWindowClosed:
self];
}
// Use shouldTerminate
```

Using `respondsToSelector:`, the application knows whether or not the delegate implements this particular method. If it does not ipmlement the method, then the default value is used. What's interesting is that the application is asking the delegate, at runtime, if it implements the method. This means that `NSApplication` does not need to be recompiled if the delegate implements the method. This runtime decision is made possible due to Objective-C's dynamic dispatch. This is not possible in static dispatched languages, like C++ or C.

Delegates and Memory Management

Just as you must unregister for notifications that you register for in your `dealloc` method, you also need to remove yourself as a delegate to objects that you are delegate for in your `dealloc` method. Classes that have delegates only keep a weak reference to their delegate to avoid a circular dependency. If you are using garbage collection, again, you do not need to worry about removing yourself as a delegate. You also can have circular references without a problem. The garbage collection system handles it all for you. You begin to see why garbage collection can save you a lot of effort.

Delegates and Notifications

One interesting side effect of setting up a delegate is that the delegate is also notified of any notification originating from the source object, if a particular delegate method is implemented. For example, `applicationWillResignActive:` is also a delegate method of `NSApplication`. So now that our controller class is the delegate for `NSApp`, we no longer need to register with the notification center to receive this notification. If we remove our notification code from `awakeFromNib` (and `dealloc`, if not using garbage collection), we will still be notified when the application becomes inactive.

Conclusion

We've covered bit more of how Cocoa applications work and how to customize them. Using actions, notifications, and delegates provides great flexibility to customize without subclassing any of Cocoa's classes. Be sure to read up on the reference documentation to see what notifications and delegates you may have available before subclassing.

About The Author

Dave Dribin has been writing professional software for over eleven years. After five years programming embedded C in the telecom industry and a brief stint riding the Internet bubble, he decided to venture out on his own. Since 2001, he has been providing independent consulting services, and in 2006, he founded Bit Maki, Inc. Find out more at <http://www.bitmaki.com/> ***and*** <http://www.dribin.org/dave/>.

Network, Server and Appliance Monitoring.

For Mac OS X

Xserve RAID

Xserve (Intel & G5)

Airport

Lithium Network Monitoring Platform

Lithium can now monitor your Xserve, Xserve RAID, Qlogic switches, Airports, Mac OS X Server... and everything else in your network.

www.lithiumcorp.com

lithium Corp.

Getting Started with REALbasic

Learning the ropes

by Norman Palardy

REALbasic is a Rapid Application Develpment (RAD) tool from REALSoftware. In previous issues we've looked at it briefly and even talked with Geoff Perlmann, the CEO of REAL Software. This month we're starting a series of articles that will focus on getting you started with REALbasic and learn how to be productive with it. For this series we'll keep current and use the latest version of REALbasic. For now, that means we'll be using REALbasic 2008r1.

Getting started

When you first start REALbasic, you see its splash screen. In figure 1, the splash screen tells me when my update plan expires. REALbasic is a subscription that entitles you to ALL of the releases while your plan is active. Since REAL has been very good about providing consistent releases in their 90 day promised time frame, every plan gets several updates while in force.

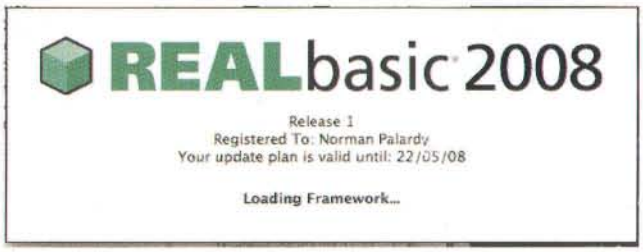


Figure 1. REALbasic splash screen

Once launched, you will immediately be looking at a new, empty project for you to start adding to.

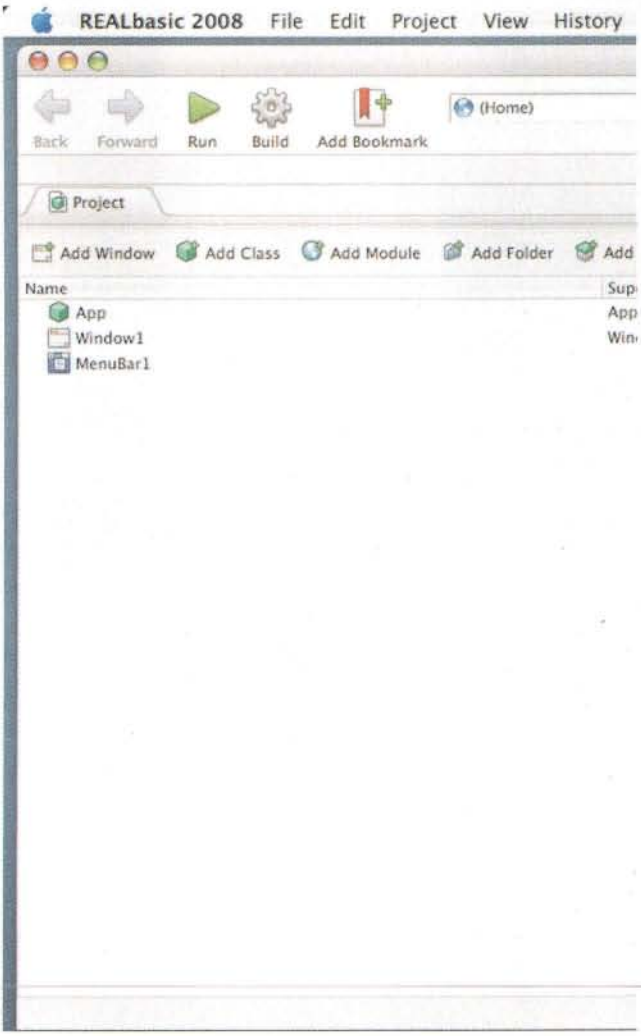


Figure 2. REALbasic default project

This default project is a fully functioning program. You could immediately run it by pressing the green Run button. It won't do much, but it does demonstrate several things that you probably want your program to have. It has a menu bar that functions and windows that you can drag around and close. The Quit menu item will quit your program as you expect.

Let's have a look at how we can add items to a window. If you double click on Window1 you should see a window like the one in figure 3.

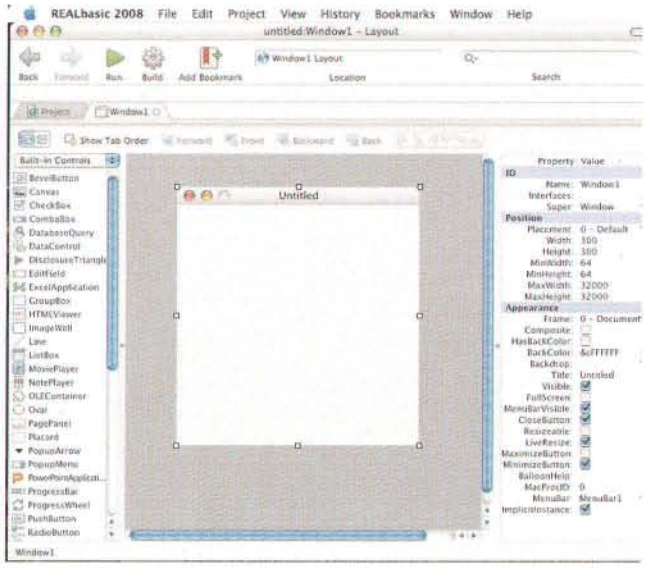


Figure 3. Editing a REALbasic window

Down the left hand side is a list of the standard controls that are available in REALbasic. Note that if you use the Personal version, this list may have fewer items.

In the center is the actual editor where you lay out the look of your window.

On the right is the properties palette that displays the properties of the currently selected item.

To add a control like a PushButton to the window you can scroll the left hand list of controls down and double click on PushButton. Alternatively, there is also a contextual menu available that permits you to add controls to a window. Some classes that have no visual representation may only be added using this mechanism.

HELLO WORLD

Usually, the first program written in a programming language to illustrate how easy it is to do something is "Hello World". In REALbasic this is not very difficult since the default project is already a running application.

Double-click on the item named Window1 and add a StaticText control to the window by selecting it from the left hand list of controls.

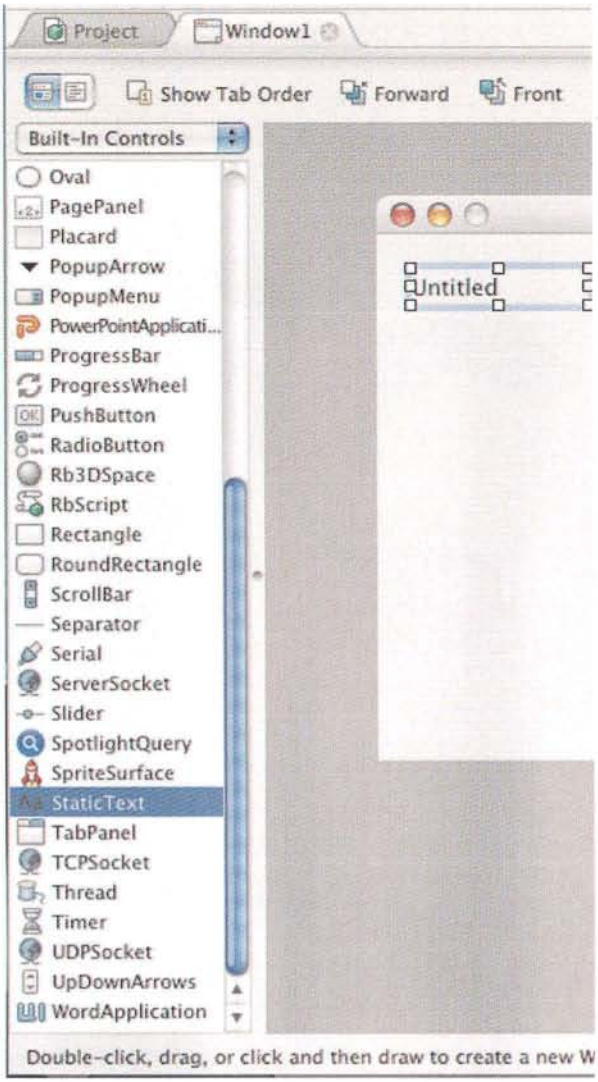


Figure 4. Control list

You'll notice that after adding the control that it remains selected and it's properties show up in the right hand property inspector. You'll see that the control has a name, StaticText1, as well as position, size and several other properties related to the appearance of the text, plus the text itself.

If you click on the ellipsis at the right hand edge of the Text property a dialog will appear where you can type in any message you want. Type in "Hello World" and close the dialog by pressing OK. Immediately in the IDE you will see that the message has changed. Now run this project by clicking the green "Run" button, and there's Hello World in REALbasic !

This window still doesn't do very much. Play around with the various properties of the StaticText control to change the font, color, bold, italic, and see how those things affect the result.

The beauty of REALbasic is that it makes trying out these kinds of changes very easy. Some changes don't even require you to run the application to see, as the IDE shows their effect immediately.

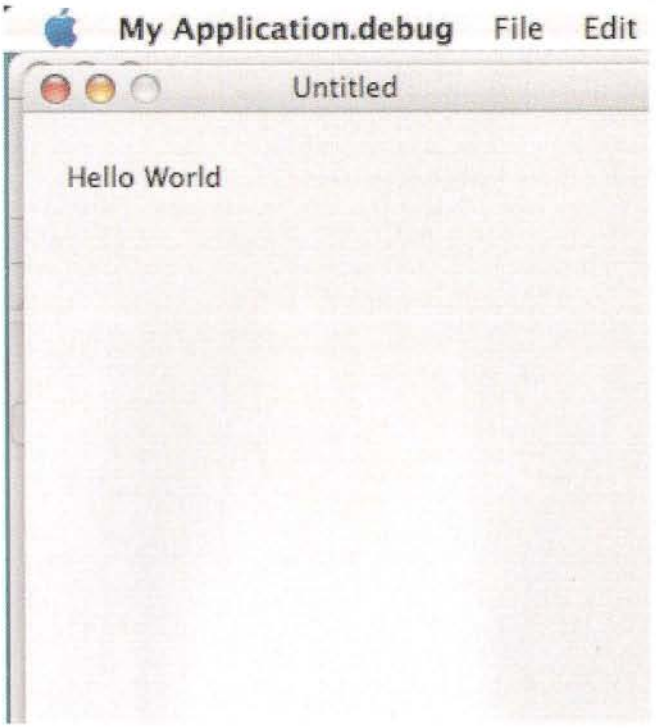


Figure 5. Hello World

What we should do is decide on an application that we want to make over a series of articles. Then, in each one, we'll add on new pieces. We'll be on our way to making the final product.

I'd like to suggest that we build an application for tracking the prices of stocks; perhaps an investment portfolio we might hold. This will involve accessing the Internet to grab quotes, graphs and a database.

We'll have to design the interface and the database as well as several windows for adding stocks to track.

Until Next Time

In the next installment we'll design the interface, the database and the windows we'll need for adding stocks to track. And we'll get started writing the basics to make the whole project come together.

Thanks for reading along. I'm looking forward to making this project come together with you over the remaining installments and introducing you to REALbasic.

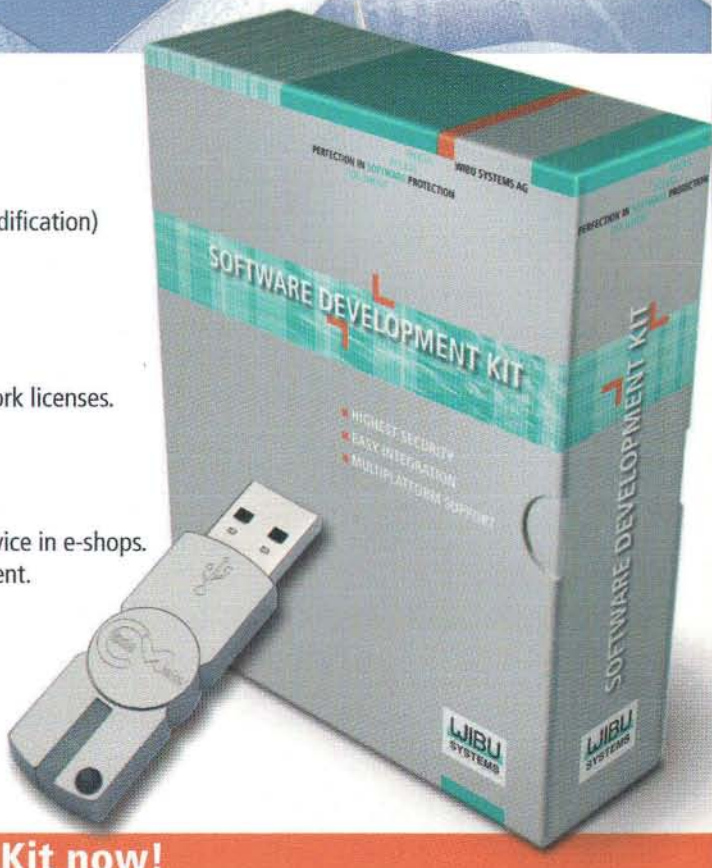
About The Author

Norman Palardy has worked with SQL databases since 1992, and has programmed in C, C++, Java, REALbasic and other languages on a wide variety of platforms. In his 15+ years of IT experience, Norman has developed innovative and award-winning applications for TransCanada Pipelines, Minerva Technologies (now XWave), Zymeta Corporation, and the dining and entertainment industry. He holds a BSc from the University of Calgary in Alberta.

CodeMeter®

No.1

in Software and Document Protection!



Highest Security

- Vendor selectable secret and private key.
- Strong encryption algorithms with AES 128-bit and ECC 224-bit.
- Best-in-class tools for automatic protection (envelope, without source modification) for Win32, Win64, .NET, Java and MacOS X Universal (PPC, Intel).

Best Flexibility

- More than 1000 independent licenses can be protected by one CM-Stick.
- One versatile hardware key for all license models including floating network licenses.
- Multi platform support including Windows, MacOS X and Linux.

New Distribution Channels

- License transfer by SOAP based CM-Talk or file based Field Activation Service in e-shops.
- Multiple-purpose, including protecting low cost software and digital content.

Unique End User Advantages

- First and smallest dongle with up to 2 Gbyte flash drive.
- No drivers necessary – can be used without administrator rights.
- CM Password Manager, secure virtual drive and secure login.

Order your Free Software Development Kit now!
Phone 1-800-6-GO-WIBU | order@wibu.us

2007 SIIA //CODiE// FINALIST
2005 SIIA //CODiE// FINALIST
2006 SIIA //CODiE// FINALIST

Mac

Universal

WIBU-SYSTEMS submitted the CodeMeter Password Manager and the CodeMeter SDK for the **Apple Design Awards 2007.**

WIBU SYSTEMS

WIBU®-SYSTEMS USA Inc.
110 W Dayton Street
Edmonds, WA 98020, USA
www.wibu.com
info@wibu.us
Tel: 1.800.6.GO.WIBU
1.425.775.6900
Fax: 1.206.237.2644

Moving Targets

Leopard's new options help you better manage mobile computers

By Greg Neagle, MacEnterprise.org

MacEnterprise.org
Mac OS X enterprise deployment project

The Future is Mobile

Each year, laptop computers comprise a larger percentage of the total number of new computers sold. This trend holds even with Apple hardware. Laptop computers can be more challenging to support in an enterprise environment than their desktop brethren, since they're not always on your network. Fortunately, with each recent major release of Mac OS X, Apple has added features that make using laptop in an enterprise environment more manageable and more secure.

OS X 10.3 "Panther" added FileVault and Mobile Accounts. FileVault enables the encryption of user data, an important feature in an enterprise environment, where the intellectual property (data) on a stolen laptop can be far more valuable than the hardware itself. Mobile Accounts allow network account information to be cached locally on the machine. This enabled people to use their network credentials to log onto their machines even when not connected to the enterprise network, and eliminated the administrative overhead of managing purely local accounts on enterprise laptops.

OS X 10.4 "Tiger" further refined mobile support with the introduction of Portable Home Directories. This addition to Mobile Accounts allowed users to synchronize a local copy of their data with a network home directory.

OS X 10.5 "Leopard" continues the refinement of mobile support. New options include the ability to create a FileVault-encrypted home directory at the same time as creating a mobile account, and External Accounts, which bring many of the features of Mobile Accounts and Portable Home Directories to removable media like FireWire drives and USB sticks.

Leopard Mobility Enhancements

An enhancement OS X administrators will appreciate is the new option to encrypt the local home directory when creating a mobile account. This saves a lot of time and makes it more likely that your users will actually have FileVault-protected accounts. Previously, in Tiger, this required two steps: first, you or the user would create the mobile account. Later, FileVault would have to be manually turned on for the newly created mobile account. This was difficult to automate, and required an admin password as well as the user's password. Even if your users had the rights to turn on FileVault for themselves, many would not. This management overhead made it far more likely that some of your mobile users were walking around with unencrypted home directories.

In Leopard, when manually creating a mobile account from the Accounts preference pane, you are given the option to protect that account with FileVault. If you choose this option, the FileVault-protected home directory is created at the same time as the mobile account.

The option to encrypt new mobile accounts is also available via the command line. Leopard includes a new mobile account tool, located in `/System/Library/CoreServices`, inside the `ManagedClient.app` bundle in `Contents/Resources`:

```
[midnight:ManagedClient.app/Contents/Resources] gneagle%
./createmobileaccount
createmobileaccount built Dec  5 2007 12:28:37
*** error: no arguments

usage: createmobileaccount -n username [-h homepath] [-P | [-p
password]] [-e] [-q] [[-x] | [-X]] [[-s] | [-S]] [-u syncURL]
[-v]
-n username : user record name
-h homepath : user home path; Default is "/Users/<username>"
-p password : user password
-P : prompt for password. A password is required for FileVault
home
-e encrypt : encrypt new home with FileVault
-q quota : max size in bytes of FileVault home
-x : create as external account on non-boot volumes. Default
-X : create as mobile account account non-boot volumes.
-s : set home sync on if home created.
-S : set home sync off if home created. Default.
-u syncURL : server target of home synchronization
-v : verbose output

Examples:
    createmobileaccount -n jsmith
    createmobileaccount -v -P -x -n jsmith
    createmobileaccount -vsdn jsmith -h /Volumes/HD3/jhome
    createmobileaccount -i

Notes:
    - createmobileaccount must run as root.
    - If you do not specify a password, the account's cached
password will be created during the account's first log in.
```

The `-e` option allows you to encrypt the local home with FileVault as the mobile account is created. The user's password is required.

Workgroup Manager/Managed Client

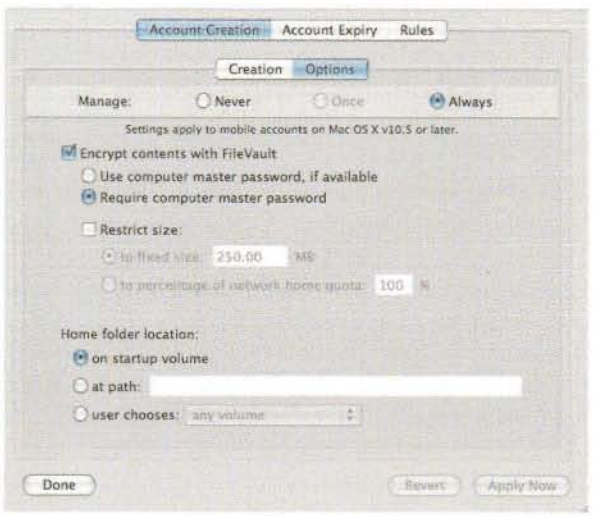


Figure 2

Perhaps of more interest to OS X administrators is the ability to enforce the option to encrypt the mobile home. This can be done via MCX using Workgroup Manager, and allows you to configure a machine so that if the user creates a mobile account, the local home is encrypted with no additional effort.

As you can see in Figure 2 (above), additional options include the ability to restrict the maximum size of a FileVault-protected home directory, and the ability to create the local home at alternate paths or on other volumes. These options are also available in the `createmobileaccount` command-line tool.

No Mobile Accounts?

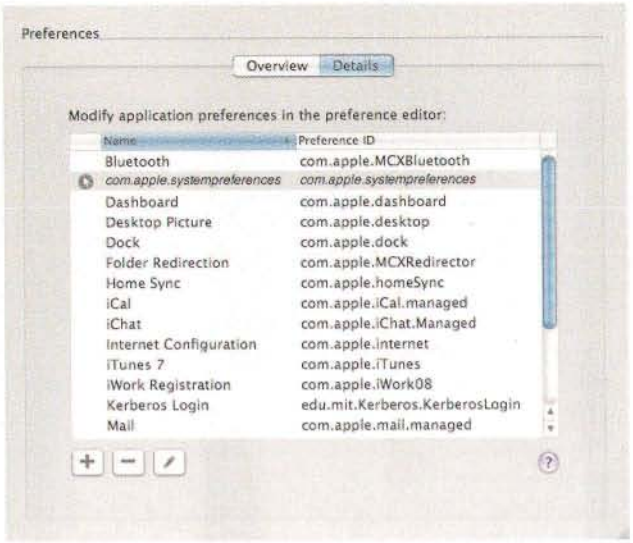
Even if you don't use mobile accounts, there are some changes in Leopard that assist with managing laptops.

When creating a local account using the Accounts pane, there is an option to protect the account with FileVault, again, saving a step. Though it is not documented by Apple, you can also enforce this setting using MCX and Workgroup Manager.

This setting is best managed at the Computer Group level. Create a Computer Group for all your laptops (or a subset of them), or use an existing group. Select the group, then click the **Preferences** icon in the Workgroup Manager toolbar.

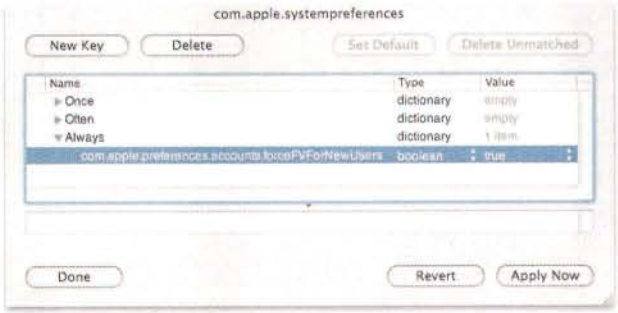
In the Workgroup Manager's Preferences editor, select the **Details** tab. Click the **+** button to add a new preference domain. Navigate to `/Applications` and double-click on the `System Preferences.app`.

The com.apple.systempreferences preference domain will be added to the Preferences Details editor, and it should look like this:



Select the com.apple.systempreferences item, and click the pencil to edit it. Delete all the imported keys using the Clear button – you don't want any of them.

Turn down the disclosure triangle next to the Always section. The New Key button should become active. Click it. A new key will be created with the name New Item. Change this to com.apple.preferences.accounts.forceFVForNewUsers. Change the type of the key to Boolean, and set its value to true. When you are done editing, it should look like this:



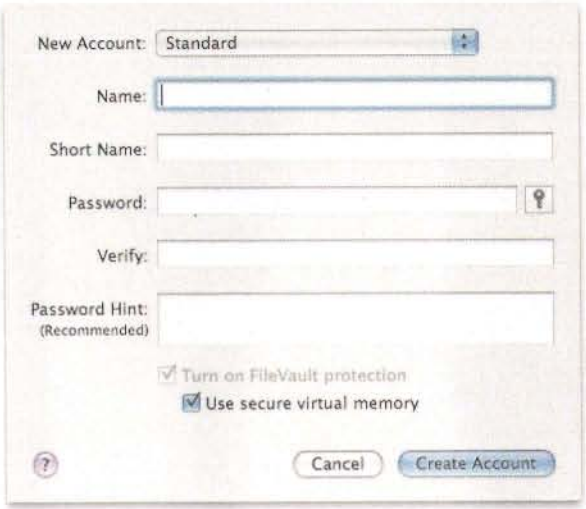
Click Apply Now to save your changes.

On a machine that is a member of the Computer Group you just edited, log in, open the Accounts preference pane, create a new account, and you'll see that Turn on FileVault protection is selected and cannot be deselected.

FileVault Improvements

Leopard also brings a couple of improvements in FileVault, which should address some of the common issues encountered when working with Apple's disk encryption technology.

Under Tiger, if a mobile user changed their network account password somewhere other than on their laptop using Apple's tools, they could lock themselves out of their FileVault-protected mobile account. Dealing with this situation could be a major pain for an OS X admin. Leopard makes the admin's (and the user's) life easier – in this situation, if the user has entered the correct network account password, but the FileVault password is different, they are prompted to enter their previous account password. This is used to unlock the FileVault home directory and update its password, again without admin intervention.



Of interest to users of both mobile and purely local accounts, the FileVault disk image format has been improved in Leopard. FileVault now uses a `.sparsebundle` disk image, rather than the `.sparseimage` format used by previous versions of OS X. The `.sparsebundle` format is much faster when recovering unused space at logout, which should make FileVault-protected users happier. Apple also claims that `.sparsebundles` are "more reliable, efficient, and scalable" than the old `.sparseimage` format.

Conclusion

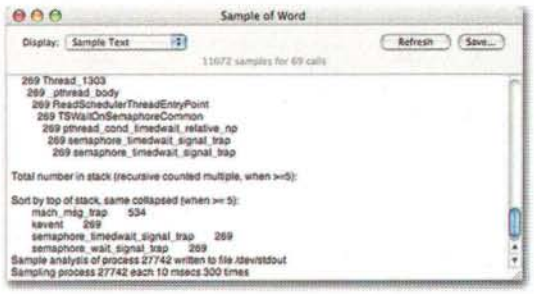
Leopard's mobility improvements, while not dramatic, nonetheless address common administrative issues. If managing mobile computers has been time-consuming in the past, Leopard's changes may help you manage laptops faster and more consistently.

About The Author

Greg Neagle is a member of the steering committee of the Mac OS X Enterprise Project (macenterprise.org) and is a senior systems engineer at a large animation studio. Greg has been working with the Mac since 1984, and with OS X since its release. He can be reached at gregneagle@mac.com.

The MacEnterprise project is a community of IT professionals sharing information and solutions to support Macs in an enterprise. We collaborate on the deployment, management, and integration of Mac OS X client and server computers into multi-platform computing environments.

Does



+

= questions?

Are you routinely looking for answers?

Imagine a whole year of answers.

MacTech Magazine is already read every month by tens of thousands of readers. Readers that represent the very heart and soul of the Mac community. Join the crowd and sign up today!

For a special one year subscription, visit:

store.mactech.com

MACTECH®

Toll Free 877-MACTECH, Outside US/Canada: 805-494-9797

Introduction to Core Animation

Using Core Animation in your Cocoa Applications

by Marcus S. Zarra

Introduction

Core Animation is arguably the jewel in the crown of OS X 10.5 Leopard. Core Animation is one of the features that Apple has been holding up as a prime reason to migrate your projects over to Leopard. In this article I am going to walk through the basics of Core Animation and how to add some simple but dazzling effects to your application.

To quote Brent Simmons from NewsGator: "Core Animation makes everything taste better. It's like hot sauce, except that everybody likes it."

Overview

Core Animation allows me to quickly and easily add visual effects to my applications that help it become more interesting and intuitive to the end-user. With Core Animation I can easily make views slide around the screen, appear, disappear, flip or just about anything else imaginable. The important part of this is that this functionality can be added very easily. It does not take an expert in Quartz or OpenGL to perform these effects.

In addition to being able to add these effects, Core Animation plays nice with the existing visual technologies. I can use Core Animation with my existing Quartz, Quicktime and OpenGL views without having to migrate everything over to one API.

But what is Core Animation? Core Animation is a set of APIs that allow developers to change UI elements over a period of time without having to generate each step in the timeline. With Core Animation I can simply tell the UI element to change x to y over n and forget about it. The API will handle all of the tweening for me.

Implicit Animations

The best way to demonstrate this is by code. In the attached project, I created a basic Cocoa application from XCode's template. The first step, after creating the XCode project was to create the application's delegate class. As is pretty standard, I named this class `AppDelegate`. The `AppDelegate` has two ivars (window and button) and one action (move):

```
@interface AppDelegate : NSObject {
    IBOutlet NSWindow *window;
    IBOutlet NSButton *button;
}

- (IBAction)move:(id)sender;
```

Once the `AppDelegate` was created I added the import statement to bring Core Animation into the class via:

```
#import <QuartzCore/CoreAnimation.h>
```

`QuartzCore.framework` is also linked into the project. From there I opened the `MainMenu.nib` file in InterfaceBuilder. In the window (Figure 1) of this application I added one button called Move!

Figure 1. The Main Window

I also wired (Figure 2) it into the AppDelegate that I also created.

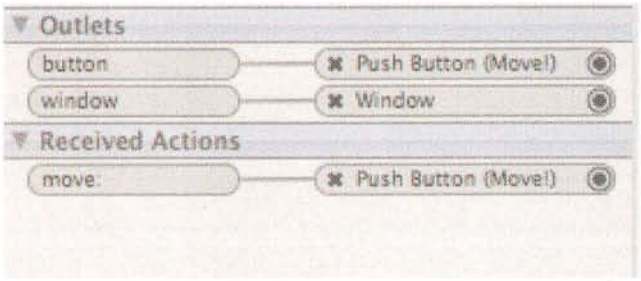


Figure 2. The button wired into the AppDelegate

With this complete in Interface Builder, it is time to add functionality to the `-(void)move:(id)sender` method to the `AppDelegate` itself:

```
- (IBAction)move:(id)sender;
{
    NSRect frame = [button frame];
    frame.origin.x -= 10;
    [[button animator] setFrameOrigin:frame.origin];
}
```

When you build and run this application, the button will gently slide across the window to the left, 10 pixels at a time.

Notice that the only special thing in this code is to make a call to the button's animator instead of the button itself. The animator is a new object included with Leopard and Core Animation. The animator is an `NSProxy` object that will accept any method call that is valid for its parent, in this case the `NSButton`. What happens when I make a call on the animator, however, is that it creates a `CABasicAnimation` in the background instead of changing the value directly on its parent. Therefore, the change is animated automatically.

Explicit Animations

But what if we want to automatically move the button back to its original location after it moves to the left? That is a slightly more complicated animation and a good excuse to build the animation manually.

When I want to do something more complex with Core Animation then I need to turn the view into a layer-backed view. The difference is that a layer-backed view has a light-weight core animation layer attached to it. When I manipulate the layer, I am indirectly manipulating the view that it is attached to.

In this example, I want to make the `NSButton` layer backed so I have added one call in the `-awakeFromNib:`

```
- (void)awakeFromNib
{
    [button setWantsLayer:YES];
}
```

This call to the button tells it to create a layer. This call only needs to be made once per view and it will automatically turn on layer backing for any subviews. Therefore, if I wanted, I could have called `-setWantsLayer:` on the `contentView` of the window instead and the button would have become layer backed as well.

Once the view is layer backed, I can grab and manipulate the layer instead of the view. Therefore, I have changed the move method as follows:

```
- (IBAction)move:(id)sender;
{
    CABasicAnimation *theAnimation = [CABasicAnimation
animationWithKeyPath:@"position.x"];
    [theAnimation setAutoreverses:YES];
    [theAnimation setToValue:[NSNumber
numberWithFloat:([[button layer] position].x - 10)]];
    [[button layer] addAnimation:theAnimation forKey:nil];
}
```

When the `-move:` method fires, I first create a new `CAAnimation` object with a key path of `position.x`. As we know from key value coding, this key path points to the x variable of the position point of the referenced object. In this case, the referenced object will be the layer backing the button I want to manipulate.

The `-setAutoreverses:` call tells the animation that I want it to reverse itself when it is done. Normally an animation would start at its `fromValue` and complete at its `toValue`. However, when `autoreverses` is set to YES the `toValue` is only the mid point — the animation completes when it has returned to the `fromValue`.

Notice that I am only calling the `-setToValue:` and I am not calling `-setFromValue:`. This is because the `fromValue` is defaulted to be the current value. Since that is what I want there is no reason to set it again.

Once I have set all of the properties on the animation that I need, I just need to add it to the layer for it to begin its work. Once I call `-addAnimation:forKey:` Core Animation will take over and perform the animation.

Animation Grouping

To further this example, I want to make the button fade out at the same time that it moves. Naturally these are two separate animations but I want them to be in perfect lock step. To accomplish this, I want to wrap them in a `CATransaction`. One thing to note: Core Animation assumes a transaction if you are performing work in the same cycle of the run loop so you don't really need a transaction for this example. However, if you are going to be doing more complex animations then it is a good habit to be in. Since `CATransaction` objects can be nested, I never know if I am going to be adding one animation effect to another call in the same run loop.

```
    [CATransaction begin];
    CABasicAnimation *moveAnimation = [CABasicAnimation
animationWithKeyPath:@"position.x"];
    [moveAnimation setAutoreverses:YES];
    [moveAnimation setToValue:[NSNumber
numberWithFloat:([[button layer] position].x - 10)]];
    [[button layer] addAnimation:moveAnimation forKey:nil];

    CABasicAnimation *fadeAnimation = [CABasicAnimation
animationWithKeyPath:@"opacity"];
    [fadeAnimation setAutoreverses:YES];
    [fadeAnimation setToValue:[NSNumber numberWithFloat:0.0f]];
    [[button layer] addAnimation:fadeAnimation forKey:nil];
    [CATransaction commit];
```

There are two major changes to this version of the `-move:` method:

First, the entire method is wrapped in `+begin` and `+commit` calls to `CATransaction`. This creates an explicit transaction around these two animations.

Second, I have added the opacity animation. Like the move animation (I renamed the animations for clarity), it animates the opacity property on the layer from its current value (which is 1.0f at this point) to 0.0f and back again.

One of the advantages of working inside of a `CATransaction` like this is that I can change values that impact the entire transaction. For example, if I want to change the duration of the animation, I can do so with one call:

```
[CATransaction setValue:[NSNumber numberWithFloat:1.0f]
                 forKey:kCATransactionAnimationDuration];
```

It should be noted that this value needs to be set before the animations are added to the layer otherwise the change will not be picked up. The reason for this is not readily apparent. When the animation gets added to the layer, the layer actually makes a copy of the animation rather than just retaining a reference. This is beneficial for many reasons including the ability to add the same animation to multiple objects.

Animating Transitions

So far I have covered implicit animations, explicit animations and animation groups. Another area of interest is animating the transition of one view to another. For instance, if I am building a wizard type dialog and I am swapping out views for the user to input data on, it is possible to animate this swap to give the user a more pleasant experience and make the transition more intuitive. To accomplish this animation, a `CATransition` animation needs to be initialized and then loaded into the `contentView` object's animation dictionary for the key "subviews". Then, whenever a `subview` is manipulated, the animation will fire.

For example, if I modified the awakeFromNib as follows:

```
- (void)awakeFromNib
{
    NSView *contentView = [window contentView];
    [contentView setWantsLayer:YES];

    CATransition *transition = [CATransition animation];
    [transition setType:kCATransitionMoveIn];
    [transition setSubtype:kCATransitionFromLeft];
    [transition setDuration:1.0f];
    [contentView setAnimations:[NSDictionary
dictionaryWithObject:transition forKey:@"subviews"]];
}
```

Any changes to the `subviews` of the `contentView` would trigger the animation I have defined. With this code in place, if I then at some later point, called:

```
- (IBAction)addPicture:(id)sender
{
    NSImageView *imageView = [[NSImageView alloc]
initWithFrame:NSMakeRect(10, 10, 50, 50)];
    [imageView setImageScaling:NSScaleToFit];
    [imageView setImage:[NSImage imageNamed:@"test1.png"]];

    [[[window contentView] animator] addSubview:imageView];
}
```

The user would see a new `contentView` appear to slide into place from the left and take the place of the current view.

Direct Layer Manipulation

In addition to being able to manipulate windows and views, it is also possible to instantiate layers directly and use them without a view attached. A layer, by itself, is a lightweight two dimensional plane in a three dimensional space. What this means is that while layers have width and height, they have no depth of their own. However, layers can have their Z-axis manipulated so that I can define the order in which layers appear within the view. Layers can also have content such as an image or text. Finally, layers have many properties that can be manipulated, such as borders, background colors, et. al.

Creating a basic layer is very simple. For example, to add a blue square to the demo project involves the following code:

```
CALayer *layer = [CALayer layer];
[layer setBounds:CGRectMake(0, 0, 150, 150)];
[layer setBackgroundColor:CGColorCreateGenericRGB(0.0, 0.0,
1.0, 1.0)];
[layer setPosition:CGPointMake(50, 50)];
[layer setZPosition:-50.0f];
[[contentView layer] addSublayer:layer];
```

First I am creating an auto-released `CALayer` with a call to `-layer`. Next, I am setting the size of the layer with a call to its bounds. Then, I am setting its background color to blue, setting its position relative to its parent, changing its Z position to -50 and finally I add it to the `contentView`.

A couple of interesting things to note here:

First, all of the manipulation of the layer uses Core Foundation references instead of `NSObject`. This was a bit confusing to me for quite a while but with the release of the iPhone SDK it has become clear. Core Animation was targeted at the iPhone from the beginning and due to the space and processing ability of the iPhone, Core Animation is meant to be as light as possible. Therefore most of Core Animation works with Core Foundations objects and references instead of `NSObject`.

Second, to add the layer to the `contentView`, I had to actually add it to the `contentView's` layer and not the view directly. Since layers are lightweight, they can only be added to other layers and not to views or windows directly.

`CALayer` objects are very flexible with regard to how they look and behave. For instance, to make the layer above more rounded with a border would only require the following changes:

```
[layer setCornerRadius:15.0f];
[layer setBorderWidth:4.0f];
```

A layer can also be made transparent, scaled, rotated and a large number of other changes. For example, it is trivial to imitate iTunes Cover Flow using `CALayer` objects.

`CALayer` objects can also be animated easily. To extend the move example above:

```
- (IBAction)move:(id)sender;
{
    [CATransaction begin];
    [CATransaction setValue:[NSNumber numberWithFloat:1.0f]
forKey:kCATransactionAnimationDuration];
    CABasicAnimation *moveAnimation = [CABasicAnimation
animationWithKeyPath:@"position.x"];
    [moveAnimation setAutoreverses:YES];
    [moveAnimation setToValue:[NSNumber
numberWithFloat:([[button layer] position].x - 10)]];
```

```
    [[button layer] addAnimation:moveAnimation forKey:nil];
    [layer addAnimation:moveAnimation forKey:nil];

    CABasicAnimation *fadeAnimation = [CABasicAnimation
animationWithKeyPath:@"opacity"];
    [fadeAnimation setAutoreverses:YES];
    [fadeAnimation setToValue:[NSNumber numberWithFloat:0.0f]];
    [[button layer] addAnimation:fadeAnimation forKey:nil];
    [layer addAnimation:fadeAnimation forKey:nil];
    [CATransaction commit];
}
```

The only changes I made to the -move: method is adding the exact same animations to the layer that I added to the button. This is possible because the layer makes a copy of the animation object instead of just a reference.

Conclusion

This is just a small taste of the power of Core Animation in Leopard. For each of the examples in this article, there are thousands of variations that are possible and probably thousands of others that no one has thought of yet.

Books and articles are just now starting to come out to cover this very exciting API and I recommend experimenting with it to discover all of its capabilities.

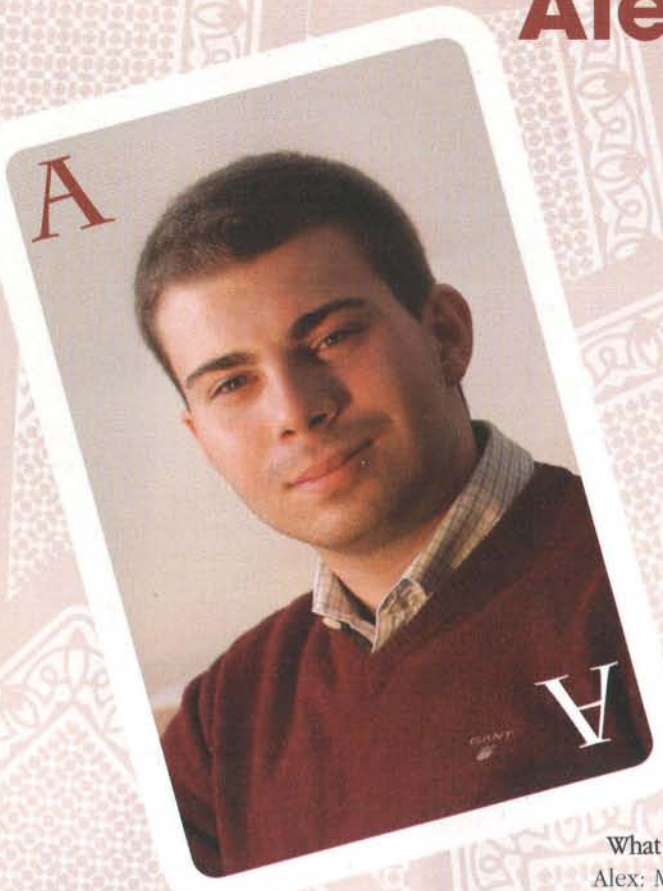
About The Author

Marcus S. Zarra is the owner of Zarra Studios, based out of Colorado Springs, Colorado. He has been developing Cocoa software since 2003, Java software since 1996, and has been in the industry since 1985. Currently Marcus is producing software for OS X. In addition to writing software, he assists other developers by blogging about development and supplying code samples.

Advertiser/Product Index

Alexander Schoen & Peter Maurer

Many Tricks



What do you do?
Alex: Creative Director, Supporter, Sales Guy, Charmer - unless I'm busy codin'
Peter: CVO (Chief Visionary Officer); Code Monkey

How long have you been doing what you do?
Alex: Working as a part of Many Tricks about a year, another year for various companies.
Peter: I started developing applications on my Atari ("first computer") back in 1990, when I was 14. Creating a full-fledged graphics editor (including a font creation tool) with Omikron Basic took me almost a year.

What was your first computer:
Alex: Macintosh Plus
Peter: Atari 1040 STFM

Are you Mac-only, or a multi-platform person?
Alex: multi-platform person - OS X, Linux, Windows

What attracts you to working on the Mac?
Alex: Performance, Security and OS X being a UNIX-derivate.
Peter: Ease of use due to the Mac's unrivaled user interface; Cocoa.

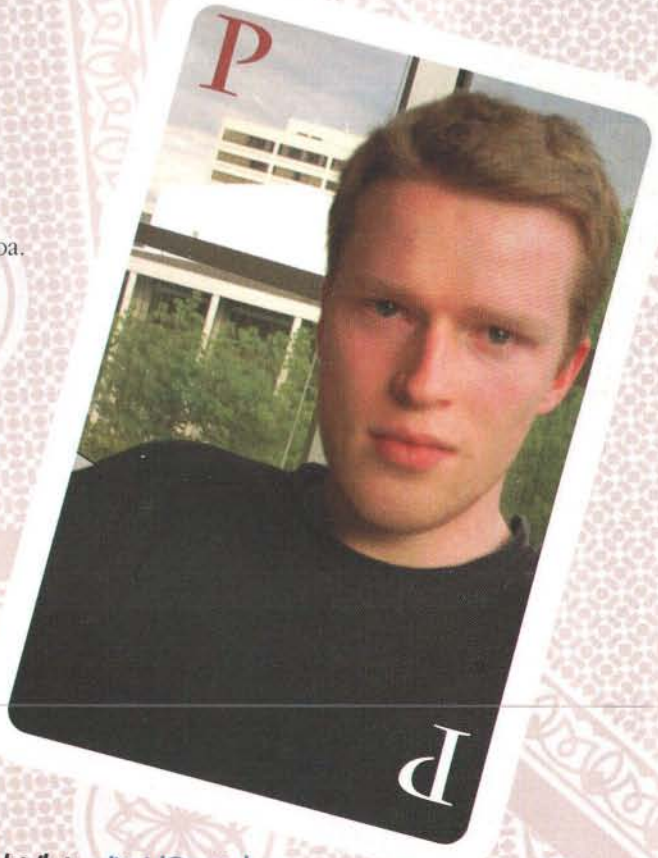
What's the coolest thing about the Mac?
Peter: The illuminated Apple logo.

Where can we see a sample of your work?
Alex: manytricks.com
Peter: On your Mac, ideally. If not, visit petermaurer.de/software for an overview of Mac projects I've been working on.

The next way I'm going to impact IT/OS X/the Mac universe is:
- Alex: Revealed secrets aren't secrets anymore. ;-)
- Peter: Mac software. Oh, wait. That's also the current way.

Any other detail you'd like us to feature?

Your company or home URL:
http://www.manytricks.com



If you or someone you know belongs in the MacTech Spotlight, let us know! Send details to editorial@mactech.com

MacBooks | SAVE up to $600

Great Gifts for Dads & Grads

At Small Dog Electronics, we carry the newest Mac products, as well as the tried-and-true models you love. Right now, we have MacBooks in stock with up to $600 off their original prices-- perfect for your favorite dad or grad. Many models are stocked with RAM, and our bundles offer the largest selection of MacBook savings around.

Visit Smalldog.com to find yours today!

Even More Reasons to Shop Smalldog.com

Many of our specials include free shipping, and every order outside of Vermont is always tax-free. Our friendly, knowledgeable staff is always ready to help with any question you may have.

Small Dog Electronics

Always By Your Side

- brand new website
- over 3,000 products
- 5-star online merchant rating
- tax-free shopping outside of VT!

www.smalldog.com **800-511-MACS** Apple Specialist